Estudi de les tècniques d'animació en HTML5

Autor: Eloi Casamayor Esteve

Director: David Sánchez Carreras

Grau en Multimèdia

1 de setembre de 2014

Contingut

1. Glossari

HTML: Llenguatge de marcat hipertext, en anglès "Hyper Text Markup Language", és el llenguatge de marques més popular per a construir documents per a la web.

DOM: Model d'Objectes del Document, en anglès "Document Object Model", és una interfície de programació d'aplicacions que proporciona un conjunt d'objectes per a representar documents HTML i XML, un model sobre com poden combinar-se entre ells i una interfície sobre com podem accedir-hi i manipular-los.

XML: Llenguatge de Marques Extensible, en anglès "Extensible Markup Language", és un llenguatge de marques desenvolupat pel W3C utilitzat per emmagatzemar dades de forma llegible.

CSS: Fulls d'estil en cascada, en anglès "Cascading Style Sheets", és un llenguatge que descriu la semàntica i presentació d'un document escrit en llenguatge de marques.

SVG: Gràfics vectorials escalables, en anglès "Scalable Vector Graphics", és una especificació basada en XML per a escriure gràfics vectorials.

GSAP: Plataforma d'animació de Greensock, en anglès "Greensock Animation Platform", és un grup d'eines per a animació que es va crear per a Flash i que ara suporta també JavaScript.

W3C: World Wide Web Consortium, consorci internacional que treballa per a desenvolupar i promocionar els estàndards per al World Wide Web.

AJAX: JavaScript Asíncron i XML, en anglès "Asynchronous JavaScript and XML" és un conjunt de tecnologies que permeten actualitzar contingut web sense haver de tornar a carregar la pàgina

2. Introducció

Un dels àmbits de coneixements que més m'agrada del grau de Multimèdia és la programació web. Tradicionalment, aquesta es divideix en el *front-end*, que es cuida de la part de software que interactua amb els usuaris, i el *back-end*, que es cuida de la part que processa la informació en el servidor. Doncs bé, en la meva petita trajectòria com a programador he trobat més gust pel *front-end*.

En els últims anys, sobretot amb el naixement de l'HTML5 i del CSS3, ha evolucionat molt ràpid la potencialitat del *front-end* i cada cop es tendeix a atribuir més feina a la màquina client. Una de les potencialitats de la interfície d'usuari és l'animació, que pot ser definida com el procés per donar sensació de moviment a imatges o objectes inanimats. Algunes de les funcions de l'animació en el web poden ser capturar la vista del visitant, donar més importància a elements de disseny pla o aconseguir una bona primera impressió als visitants.

Al llarg dels quatre cursos de Multimèdia hem tractat molt per sobre les diferents opcions i tècniques per a desenvolupar animacions web. Amb aquest treball, doncs, he volgut cercar les tecnologies existents d'aquest àmbit i aplicar-les en una animació web interactiva.

El present és un treball d'aplicació de les eines d'animació d'HTML5 per al desenvolupament d'un nou projecte interactiu. La investigació i la recerca han tingut un paper important. He volgut ser pioner en l'aplicació de les tècniques i mètodes emprats. En aquest sentit, el treball ha estat un repte. Un repte amb el qual he volgut provar i demostrar la potencialitat de l'HTML5 i aprendre no només a fer una animació, sinó investigar les tecnologies per a crear-la. També he pretès amb la meva dedicació contribuir a potenciar l'animació en HTML5.

2.1 Objectius generals

- Cercar els mètodes existents actualment per a crear animacions en HTML5.
  Per tal d'aprofundir els coneixements que tinc sobre com crear animacions HTML5, cal investigar les possibilitats que ofereixen les tecnologies SVG, Canvas, CSS3 i JavaScript. També és imprescindible conèixer les eines existents, tant programari com llibreries.
- Realitzar una animació a partir d'una de les tecnologies o una barreja d'aquestes.

La tria de l’eina i tecnologia a utilitzar ha de tenir com a criteri principal que sigui òptima per al desenvolupament d’una animació web interactiva per a dispositius mòbils. La realització de l’animació ha de constar d’un *briefing* on s’estableixin les característiques principals, un guió o esquema, el disseny dels escenaris i personatges i el desenvolupament final.

3. Nucli de la memòria

3.1 Antecedents

Es poden trobar a la xarxa gran diversitat d'animacions HTML5 amb diversos nivells de complexitat i resultats diferents. Després d'una recerca àmplia n'he triat quatre de representatives que estan ben documentades en quant als mètodes i tècniques de desenvolupament.

THE GOOD MAN

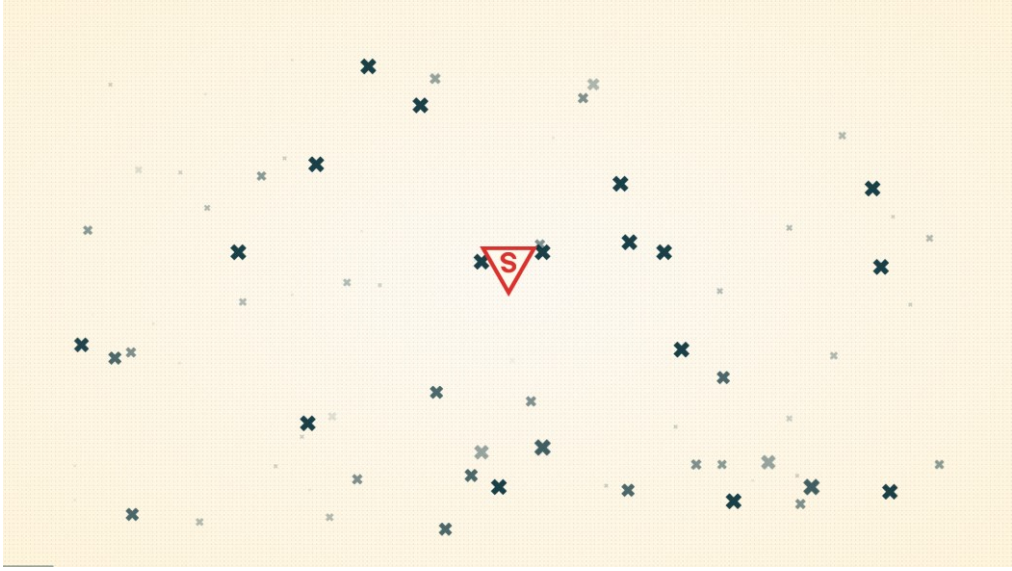


Figura 1: captura de l'animació "The Good Man"

Publicació: gener 2013

Autor: Pedro Ivo Hudson

The Good Man és una transcripció visual d'una narrativa. Basada en elements HTML animats amb CSS3 i sense utilitzar imatges, aquesta animació es veu perfectament suau en navegadors moderns, és escalable i presenta un disseny molt acurat. El seu autor, Pedro Ivo Hudson, pretén que aquest treball, que va ser el seu projecte de graduació en Disseny Gràfic, ajudi a construir l'animació en els navegadors web. Aquesta animació web ha sigut reconeguda amb premis molt importants com els AWWWARDS i FWA.

MADMANIMATION

Figura 2: captura de l’animació “Madmanimation”

Publicació: abril 2011

Autors: Anthony Calzadilla, Andy Clarke, Geri Coady.

És una animació inspirada en els crèdits d’apertura de la pel·lícula *Mad Men*. Està creada a partir d’imatges PNG i animada amb CSS3 i JavaScript. L’autor explica que el codi es basa en una llista de 16 ítems, una per a cada escena. En cada ítem hi ha tot l’HTML i les imatges d’una escena i per defecte tots els ítems se’ls aplica l’ordre “display: none”. Mitjançant jQuery s’aplica una classe “go” a un element i s’activa l’animació CSS3 assignada a l’escena. Quan s’ha acabat el temps, s’elimina la classe, es para l’animació i s’aplica la classe “go” al següent element de la llista, començant de nou la cadena. L’autor admet que utilitza codi HTML no semàntic però ho justifica assegurant que HTML no està pensat perquè se’n faci aquest ús.

The man from Hollywood

Figura 3: captura de l’animació “The man from Hollywood”

Publicació: abril 2010

Autor: Tyler Gaw

És una animació que fa servir la tècnica de tipografia cinètica basada en el diàleg d’una escena de la pel·lícula *Four Rooms*. El seu autor assegura que només utilitza HTML, CSS i JavaScript; cap imatge. Segons explica, les animacions estan fetes amb transicions i animacions CSS i el JavaScript és usat simplement per a canviar els noms de les classes en el temps apropiat.

Run Pixie Run

Figura 4: captura del joc “Run Pixie Run”

Publicació: febrer 2013

Aurtor: Goodboy

Es tracta d’un joc en què la única interacció de l’usuari és clicar (o tocar la pantalla) per agafar objectes bons, que sumen punts, i evitar els dolents, que maten el personatge. Aquest joc utilitza la tecnologia Canvas i ha sigut desenvolupat amb el motor de renderitzat pixi.js. Els autors, una empresa de disseny web, el va crear per donar a conèixer el seu motor de renderitzat de Canvas, pixi.js.

Dream On

Figura 5: captura de l'animació "Dream On"

Publicació: 31 gener 2014

Autor: Soleil Noir

És una animació feta per Soleil Noir, una agència de publicitat i disseny web francesa. Es basa en 10 "caselles" en cadascuna de les quals hi ha una frase i formes animades. Per navegar entre elles, es pot fer mitjançant *scroll* o a través d'uns enllaços a la part dreta. Les animacions són en tres dimensions i són molt suaus. Els autors han fet servir les llibreries GSAP, Pixi i ThreeJS per a desenvolupar l'animació web, que utilitza Canvas i WebGL. Va guanyar el premi "Site of the day" de la FWA el 24 de febrer del 2014.

Valoració dels antecedents

Cap al 2010 les animacions HTML5 eren proves de desenvolupadors o prototips tecnològics. En canvi, en l'actualitat l'animació és un element amb molta importància en els webs més moderns. Si ens fixem en els projectes guanyadors dels principals concursos de disseny web, on hi ha sempre reflectides les tendències tecnològiques i de disseny, veiem que en tots hi té un paper destacat. Per tant podem dir que l'animació web està en auge.

He comprovat que les animacions que he seleccionat es veuen perfectament en dispositius mòbils. Totes excepte "The man from Hollywood", en la qual falla la sincronització de l'àudio i l'animació. Actualment és impensable un projecte web que exclogui els mòbils i tauletes ja que el transit mòbil és cada cop més gran. Segons un estudi del PewResearch Internet Project basat en els usuaris d'Estats Units, el temps invertit dels usuaris amb el transit mòbil (contant el transit en aplicacions, no només el navegador) superava el gener del 2014 el temps invertit en transit en ordinadors i portàtils. Això m'ha fet decidir de prioritzar al màxim la correcta visualització de la meva animació en mòbils i tauletes.

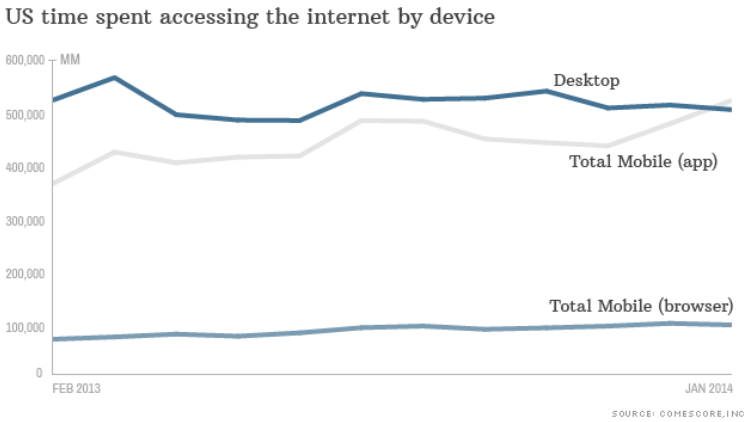


Figura 6: gràfic que mostra el temps invertit en transit mòbil i trànsit d'ordinadors de sobretaula als EUA. Font: CNN.com

Un aspecte que valoro de "Run Pixie Run" i de "Dream On" és que ocupen la finestra completa. Això facilita una major immersió de l'usuari i una major concentració. Ho he implementat a la meva animació, juntament amb la opció de veure-la a pantalla completa. A part, també valoro positivament el fet que siguin animacions interactives.

3.2 Preproducció

3.2.1 Planificació

a. Planificació inicial

La planificació inicial contemplava els següents set punts:

1. Diagrama de Gantt
2. Estudi de les noves tecnologies, eines i programes que permeten fer animacions
3. Realització de l'escaleta, l'*storyboard*, el guió tècnic i el guió literari de l'animació.
4. Disseny gràfic.
5. Tria i estudi de la tècnica més adient i desenvolupament de l'animació.
6. Entrega
7. Exposició

I aquestes fases les vaig repartir en el temps de la següent manera.

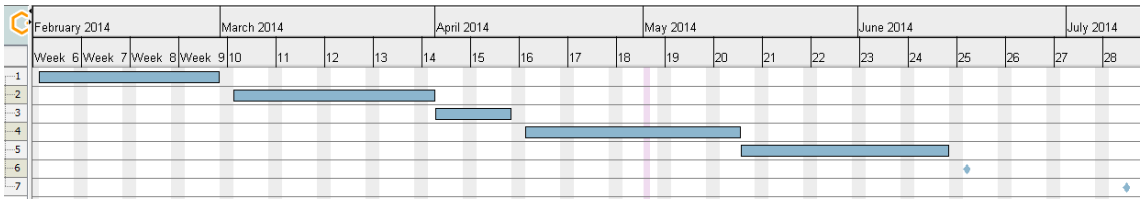


Figura 7: diagrama de Gantt de la planificació inicial

Tot i que em va servir per veure la magnitud del treball i per establir certs terminis temporals, no m'ha sigut possible seguir aquesta planificació inicial.

Dos factors han fet alterar molt les fases que havia planejat. El primer factor ha estat que la investigació i recerca m'ha portat molt més temps del que em pensava. Vaig començar al febrer i al maig encara estava en aquesta primera fase. L'altre factor ha estat que, un cop havia començat una animació i havia invertit moltes hores, vaig decidir començar-la de nou redefinint-la completament.

b. Planificació segona

En data a 5 de maig de 2014, vaig fer una segona edició de la planificació. La investigació de les tecnologies, eines i antecedents m'havia costat molt més temps del que havia planificat inicialment. Per això vaig haver de retallar temps en les tasques posteriors de desenvolupament i memòria.

Les tasques i subtasques quedaven de la següent manera:

1. Cerca d'antecedents
2. Investigació de tecnologies i eines
   - 2.1. Recerca de les tecnologies
   - 2.2. Esquema i redacció del resultat de la cerca les tecnologies
   - 2.3. Recerca de les eines
   - 2.4. Redacció del resultat de la cerca de les eines
3. Triar la tècnica, estudiar-la i provar-la
4. Guió i disseny de l'animació
   - 4.1. Disseny Gràfic
   - 4.2. Estudi didàctic
   - 4.3. Disseny dels nivells
5. Desenvolupament de l'animació
   - 5.1. Desenvolupament introducció
   - 5.2. Desenvolupament escenari del vestuari
   - 5.3. Desenvolupament joc 1
   - 5.4. Desenvolupament joc 2
   - 5.5. Desenvolupament joc 3
   - 5.6. Desenvolupament joc 4
6. Redacció de la memòria i acabats
   - 6.1. Preparar exposició
7. Entrega
8. Exposició

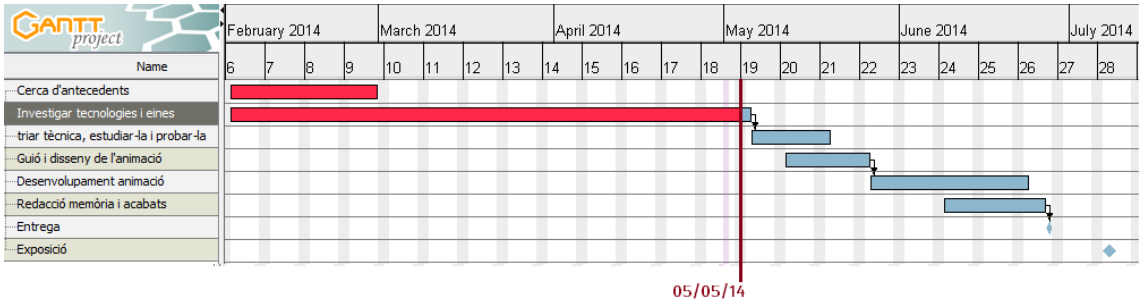


Figura 8: diagrama de Gantt de la planificació segona

Els passos 3 i 4 se m'han allargat molt més en el temps, principalment perquè vaig iniciar una animació i després la vaig desestimar i vaig començar-ne una altra. Aquest fet, sumat al llarg temps que m'havia portat la investigació, em va portar a haver de demanar una pròrroga per tal de poder entregar el treball més tard. Inicialment l'entrega establerta era el 27 de juny.

c. Planificació final

En data de 15 de juny, un cop vaig saber que m'havien acceptat la pròrroga d'entrega, vaig modificar la planificació per segona vegada modificant els temps de les últimes fases.

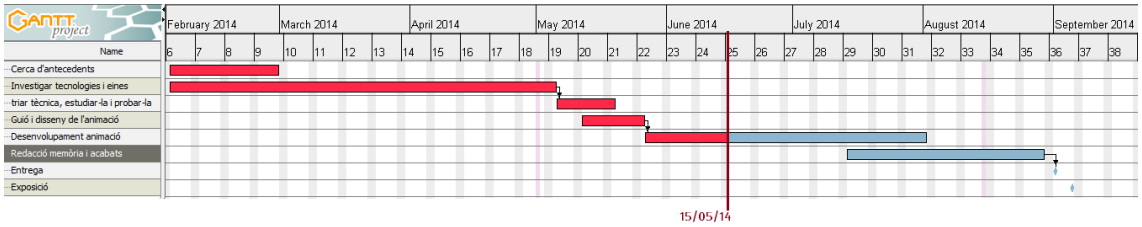
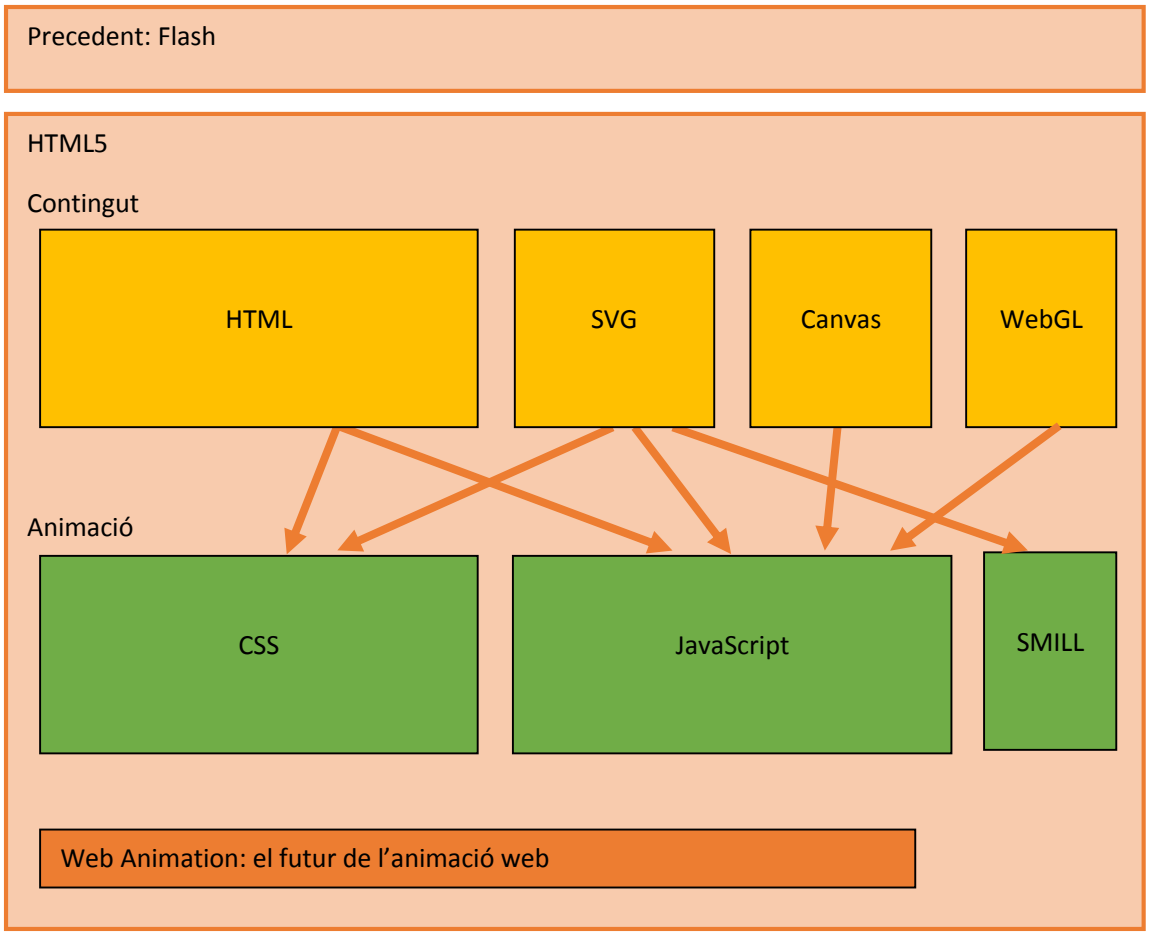


Figura 9: diagrama de Gantt de la planificació final

d. Calendari de treball

Tasca	Inici	Final
1. Cerca d'antecedents	03/02/14	28/02/14
2. Investigació de tecnologies i eines	03/02/14	04/05/14
1.1. Recerca de les tecnologies	03/02/14	03/03/14
1.2. Esquema i redacció del resultat de la cerca les tecnologies	03/03/14	10/03/14
1.3. Recerca de les eines	10/03/14	28/04/14
1.4. Redacció del resultat de la cerca de les eines	28/04/14	04/05/14
3. Triar la tècnica, estudiar-la i provar-la	04/05/14	25/05/14
4. Guió i disseny de l'animació	12/05/14	26/05/14
1.1. Disseny Gràfic	12/05/14	15/05/14
1.2. Estudi didàctic	12/05/14	22/05/14
1.3. Disseny dels nivells	15/05/14	26/05/14
5. Desenvolupament de l'animació	26/05/14	01/08/14
1.1. Desenvolupament introducció	26/05/14	08/06/14
1.2. Desenvolupament escenari del vestuari	08/06/14	22/06/14
1.3. Desenvolupament joc 1	22/06/14	02/07/14
1.4. Desenvolupament joc 2	02/07/14	13/07/14
1.5. Desenvolupament joc 3	13/07/14	24/07/14
1.6. Desenvolupament joc 4	24/07/14	01/08/14
6. Redacció de la memòria i acabats	17/08/14	31/00/14
1.1. Preparar exposició	25/08/14	31/08/14
7. Entrega	01/09/14	
8. Exposició	05/09/14	

3.2.2 Recerca de les tecnologies per a l'animació web



El Flash: precedent de l'animació HTML5

SWF (Small Web Format) és un format de gràfics vectorials creat per Macromedia. Els fitxers poden ser creats amb Adobe Flash o amb molts altres programes i poden ser executats per un navegador web mitjançant el plugin Adobe Flash Player. Aquest plugin és molt utilitzat per a reproduir vídeos a la web i alguns navegadors, com el Google Chrome, el tenen integrat. Els fitxers SWF també poden ser reproduïts autònomament si estan encapsulats. Es tracta d'un format d'animació vectorial, però també admet mapes de bits (o *bitmaps*), que són matrius rectangulars de píxels.

El format Flash admet des de la versió 5 la inclusió de programació amb el llenguatge ActionScript. Aquest és un llenguatge expressament creat per a crear aplicacions Flash interactives i ha evolucionat fins a la versió 3.0, que és l'actual i que permet programació orientada a objectes.

Si parlem del Flash com a antecedent de l'animació HTML5 és perquè va ser molt utilitzat fa un temps però ara tant el format SWF com el llenguatge ActionScript s'estan deixant d'utilitzar en tots els àmbits. Els motius són els següents:

- És dolent per a l'accessibilitat dels webs. Tot i que va evolucionar amb el temps cap a millor, flash no està pensat per a crear webs i aplicacions accessibles per a persones amb dificultats visuals, auditives, etc. L'HTML5, en canvi, prioritza l'accessibilitat. Tant és així, que hi ha una recomanació oficial del W3C anomenada "Directrius per a l'accessibilitat del contingut web" i també un programari on-line per a validar el codi HTML.
- El plugin Adobe Flash Player és insegur. Des de que existeix, s'han trobat forats de seguretat importants que han fet perillar la seguretat dels usuaris.
- És un format propietat d'una empresa: Adobe Systems. Per tant, aquesta empresa té control total sobre què fa o deixa de fer amb el format, i el futur que li dóna. El plugin Adobe Flash Player és d'ús gratuït.
- Implica pèrdua de control per part de l'usuari. El navegador interpreta l'arxiu SWF com un únic element, per la qual cosa no té control sobre cada element interior, com passa per exemple amb el Canvas.

HTML5

HTML5 (en anglès, HyperText Markup Language 5) és la cinquena versió del llenguatge bàsic de la web. És un estàndard obert i està regulat pel Consorci W3C.

Dins el nom HTML5 s'hi inclou moltes APIs i noves eines, a part del llenguatge HTML en sí, que suma moltes etiquetes noves. Permet llegir elements Canvas, SVG, àudio i vídeo directament. Algunes de les APIS noves són la de treball off-line, geoposicionament, emmagatzematge, WebSockets.

De tots els elements integrats dins de l'HTML5 implicats en l'animació web, els he estructurat de la següent manera:

- Elements per a construir el contingut de la pàgina:
    - HTML
    - SVG
    - Canvas
    - WebGL
- Elements per a animar el contingut:
    - CSS3
    - JavaScript: la funció webAnimationFrame()
    - SMILL

Bàsicament els mètodes d'animació HMTL5 són l'ús de JavaScript (anomenat animacions imperatives) i l'ús de CSS3 (anomenat animacions declaratives). Simplificant al màxim, podem dir que s'utilitza CSS per a les animacions més simples i JavaScript quan volem tenir efectes avançats com el rebot, interacció amb el ratolí, pausa, etc.

Les animacions imperatives (JavaScript) s'executen al fil principal del navegador. Precisament això pot ser un inconvenient perquè comporta que en els moments en què el fil principal està sobre ocupat, es salta fotogrames i aleshores l'animació no és suau. Per altra banda, l'avantatge és que amb JavaScript tenim un control molt gran sobre l'animació ja que interacció entre animacions, pausa, començament, encadenament... són tasques fàcils de fer.

Les animacions declaratives són les transicions i animacions CSS. L'avantatge d'aquest tipus d'animacions és que el navegador les pot optimitzar creant capes de renderitzat i no li cal realitzar totes les tasques en el fil principal. El gran desavantatge de les animacions declaratives és que és el desenvolupador no té tant control sobre les animacions com en les imperatives. És molt difícil poder executar seqüències d'animacions en CSS i el codi de les animacions arriba a ser molt complex.

Elements per a construir el contingut de la pàgina

- HTML

És el llenguatge de marques en sí. És l'estàndard de referència per a la construcció de contingut en una pàgina web, i ho fa seguint les regles del DOM ("Document Object Model"). Basa la seva filosofia de desenvolupament en la referenciació, vol dir que qualsevol element que no sigui text o etiquetes HTML s'afegeix fent una referencia a la ubicació de l'element mitjançant text. El W3C, qui controla aquest estàndard, es preocupa molt per l'accessibilitat web i estableix que tot disseny HTML ha de complir unes Pautes d'Accessibilitat al Contingut Web.

- SVG

Scalable Vector Graphics (SVG) és una família d'especificacions basada en XML per descriure gràfics vectorials bidimensionals, tant estàtics com dinàmics. Es tracta d'un estàndard obert desenvolupat i recolzat pel W3C des del 1999. HTML5 l'incorpora com a una nova etiqueta, gràcies a la qual podem incloure codi SVG dins dels arxius HTML.

És renderitble per tots els navegadors moderns (incloent Internet Explorer a partir de la versió 9.0) sense necessitat d'extensions. Es basa en objectes, que poden ser de tres tipus: gràfics vectorials, gràfics raster i text. La versió actual és la 1.1 (del 2001) i la 1.2 és encara esborrany. L'SVG és totalment transparent pel navegador: el navegador interpreta cada element per separat , d'una manera semblant a la que ho fa amb el DOM d'HTML. A més a més, pot ser barrejat amb altres continguts HTML.

Al ser XML, és un llenguatge de marques i pot ser estilitzat amb un full d'estil CSS, com l'HTML. Els elements SVG poden ser tractats per un llenguatge script, la qual cosa ens permet afegir-li interacció amb l'usuari. Es poden crear arxius SVG amb editor de text però també amb programes de dibuix especialitzats com per exemple *Adobe Illustrator* o *Inkscape*.

La W3C estableix 3 formes d'animar SVG:

- Utilitzant els elements d'animació que incorpora SVG. La mateixa especificació ens proveeix d'etiquetes per a descriure modificacions dels elements del document al llarg del temps.
- Mitjançant JavaScript, podem modificar els elements del DOM com ho podríem fer amb HTML.

- Utilitzant SMIL.

SVG és compatible amb quatre elements definits a l'especificació d'animació SMIL. Aquestes etiquetes son: '*animate*', '*set*', '*animateMotion*' i *AnimateColor*. A més a més, inclou més etiquetes, també compatibles amb SMIL: '*animateTransform*', '*path*', '*mpath*', '*keyPoints*' i '*rotate*'.

Exemple de codi d'animació SVG:

```
<rect id="RectElement" x="300" y="100" width="300" height="100"
fill="rgb(255,255,0)"  >

    <animate attributeName="x" attributeType="XML" begin="0s" dur="9s"
fill="freeze" from="300" to="0" />

    <animate attributeName="y" attributeType="XML" begin="0s" dur="9s"
fill="freeze" from="100" to="0" />

    <animate attributeName="width" attributeType="XML" begin="0s" dur="9s"
fill="freeze" from="300" to="800" />

    <animate attributeName="height" attributeType="XML" begin="0s" dur="9s"
fill="freeze" from="100" to="300" />

  </rect>
```

- Canvas

És un element HTML incorporat en l'última versió (HTML5) que permet la generació de gràfics 2D dinàmicament per mitjà d'*scripting*. Va ser començat a desenvolupar per l'empresa *Apple* el 2004 i va ser adoptat com a estàndard pel WHATWG el 2006. L'especificació definitiva encara no està publicada, de moment és un esborrany. És suportat per tots els navegadors moderns, incloent Internet Explorer des de la versió 9.0. La principal diferència amb l'SVG és que es basa en un mapa de bits, no en un mapa vectorial, com en el cas de l'SVG.

L'objecte <canvas> té només dos atributs, *with* i *height*. El llençol (superfície del canvas) és una matriu amb origen 0,0 a l'extrem superior esquerra i que creix cap a la dreta i cap avall. Igual que l'SVG, el canvas pot ser accedit mitjançant JavaScript, la qual cosa ens permet desenvolupar aplicacions interactives i animacions entre d'altres. De fet ha de ser manipulat amb JavaScript, ja que el canvas en sí és només el llençol, sense gràfics. L'especificació Canvas defineix diferents mètodes JavaScript per a dibuixar línies, rectangles, cercles, texts i per afegir imatges.

Al ser un mapa de bits, cada element del Canvas no és independent. Si volem variar un sol element, s'ha de redibuixar tot el llençol. Així mateix, al no ser els elements independents un de l'altre, si volem manegar un esdeveniment de JavaScript per clicar-lo, per exemple, haurem de calcular la coordenada que ocupa mitjançant JavaScript.

Per animar amb Canvas, es crea una funció JavaScript que "refresqui" el llençol tantes vegades per segon com vulguem, per exemple utilitzant setInterval. Aleshores, dins d'aquesta funció anem creant elements canvas, mitjançant els mètodes definits. Alguns d'aquests mètodes són:

- *fillRect(x, y, w, h)*: per dibuixar un rectangle.
- *strokeRect(x, y, w, h)*: per dibuixar el perímetre d'un rectangle.
- *clearRect(x, y, w, h)*: per borrar els píxels que queden dins del rectangle
- *arc(x, y, radius, startAngle, endAngle, anticlockwise)*: per dibuixar un arc o circumferència.

Exemple de codi d'animació Canvas:

```
<canvas id="myCanvas" width="200" height="100" style="border:1px solid
#000000;">
</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.fillStyle = "#FF0000";
ctx.fillRect(0,0,150,75);
</script>
```

- WebGL

WebGL és una especificació estàndard dissenyada per a mostrar gràfics 3D dins de Canvas en navegadors web nativament (sense necessitat d'un plug-in) i aprofitant l'acceleració per hardware (GPU). Va ser creat per la Fundació Motzilla i des del 2006 és desenvolupat per la organització sense ànim de lucre *Kronos Group*. Està basat en OpenGL 2.0 i és suportat per tots els navegadors moderns, tant d'ordinador com mòbils.

Està dissenyada per treballar directament amb la unitat de processament gràfic (GPU), i utilitza un llenguatge de baix nivell. Hi ha moltes llibreries que fan molt més fàcil la programació amb WebGL, com per exemple Three.js. Es poden crear escenes amb programes de creació 3D, sense necessitat de programar, i exportar-les a WebGL.

Elements per a animar el contingut de la pàgina

- CSS3

Cascading Style Sheets (CSS, Fulls d'Estil en Cascada) és el llenguatge utilitzat per donar estil a un llenguatge de marques. L'última versió de CSS (CSS3) incorpora moltes novetats, agrupades en mòduls. Dos d'aquests mòduls són les transicions i animacions.

Les transicions CSS són efectes que permeten que un element canviï gradualment d'un estil a un altre. Per definir-les, s'ha d'especificar la propietat que es vol afectar i el temps de duració. L'efecte de transició comença quan la propietat especificada canvia. Per exemple, fer que comenci quan el ratolí és sobre l'objecte.

Exemple de codi de transició CSS3:

```
div {   transition: width 2s;}

div:hover {
    width: 300px;
}
```

Les animacions CSS3 són una mica més complexes. Amb aquestes podem animar objectes fent que canviïn gradualment entre dos o més estats. Cal especificar la duració de l'element, i dins l'animació, els diferents estadis pels que passarà l'element, establint un percentatge del temps en cadascun i les propietats que volem d'estil per a cada estadi.

A més a més tenim més propietats per a establir un retard (*animation-delay*), la direcció (*animation-direction*), un estil per quan està parada (*animation-fill-mode*), el número d'iteracions (*animation-iteration-count*), el nom de l'animació (*animation-name*), l'estat de l'animació (*animation-play-state*) i la corba de velocitat de l'animació (*animation-timing-function*).

Exemple de codi d'animació CSS3:

```
div {
    animation: myfirst 5s;
}

@keyframes myfirst {
    0%   {background: red; left:0px; top:0px;}
    25%  {background: yellow; left:200px; top:0px;}
    50%  {background: blue; left:200px; top:200px;}
    75%  {background: green; left:0px; top:200px;}
    100% {background: red; left:0px; top:0px;}
}
```

- SMIL

SMIL (*Synchronized Multimedia Integration Language*) és un llenguatge obert i gratuït basat en XML ideat expressament per a desenvolupar presentacions multimèdia interactives. És un estàndard de la W3C definit en l'especificació "SMIL Animation", del 2001. Els objectius d'aquesta especificació són definir un llenguatge basat en XML que permeti als autors crear presentacions multimèdia i facilitar la reutilització de la semàntica de SMIL en altres llenguatges basats en SML. Per exemple, SML pot integrar-se amb XHTML i amb SVG. Tot i les intencions dels seus creadors, el seu ús no és estès entre els programadors web i ha quedat arraconat d'entre tecnologies d'animació web.

SMIL està dissenyat per descriure, a través d'etiquetes XML un seguit d'elements:

- Fonts de contingut, que poden ser imatges estàtiques, àudio, vídeo, text pla, fluxos de text i animacions.
- Sincronització de les fonts de contingut, que es poden reproduir alhora o en seqüència.
- Temporalització. Podem indicar quan volem que comenci i pari una font de contingut, mitjançant un cronòmetre o mitjançant esdeveniments script.
- Posició de les fonts de contingut, tamany, rotació...
- Enllaços per interaccionar amb l'usuari
- Animacions de les propietats dels objectes:

- Els atributs numèrics d'un element
- Els atributs de transformació (translació, rotació)
- Els atributs de color
- Una trajectòria de moviment

- JavaScript: l'API requestAnimationFrame()

És una interfície de programació d'aplicacions (API) JavaScript pensada per animacions, que permet fer canvis a la pàgina cada pocs mil·lisegons. Pretén ser substituïdes per les animacions basades en les funcions setTimeout i setInterval. És recomanada per la W3C en l'especificació "Timing control for script-based animations" de l'octubre del 2013.

Aquest mètode s'encarrega de decidir la "freqüència de repintat" òptima per a cada agent d'usuari, la qual cosa és un gran avantatge respecte l'ús de les funcions setTimeOut i setInterval. Aquesta funció optimitza les animacions estalviant molta feina al navegador. Permet optimitzar animacions simultànies en un sol flux de repintat, i també incloure-les en el flux de les transicions i animacions CSS o SVG SMIL. A més a més, quan l'animació no és visible en la pantalla, no s'executa i ens permet estalviar feina i consum de la CPU.

Exemple d'animació a partir de l'API JavaScript requestAnimationFrame():

```
<!DOCTYPE html>

<title>Script-based animation using requestAnimationFrame</title>

<style>

div { position: absolute; left: 10px; padding: 50px; background: crimson;
color: white }

</style>

<script>
```

```
var requestId = 0;
function animate(time) {
  document.getElementById("animated").style.left =
    (time - animationStartTime) % 2000 / 4 + "px";
  requestId = window.requestAnimationFrame(animate);
}
function start() {
  animationStartTime = window.performance.now();
  requestId = window.requestAnimationFrame(animate);
}
function stop() {
  if (requestId)
    window.cancelAnimationFrame(requestId);
  requestId = 0;
}
</script>
<button onclick="start()">Click me to start!</button>
<button onclick="stop()">Click me to stop!</button>
<div id="animated">Hello there.</div>
```

Font del codi: www.w3.org

Web animation: el futur de l'animació web

És una nova especificació per a contingut animat al web. El primer esborrany públic es publicà el 25 de juny del 2013 i l'últim publicat data del juny del 2014. Aquesta especificació defineix un model de sincronització i temporalització dels canvis a una pàgina web i s'està elaborant conjuntament amb el grup de treball de CSS i el de SVG.

Pretén definir un model d'animació abstracta que reculli les característiques comunes de les transicions CSS3, les animacions CSS3 i SVG i que puguin ser definides a partir de llenguatge script.

L'objectiu d'aquesta especificació és, doncs, ajuntar els dos tipus d'animacions (imperatives i declaratives) per mantenir els avantatges de les dues.

3.2.3 Recerca d'eines per fer animacions HTML5

Hi ha dos grans grups d'eines existents per a desenvolupar animacions HTML5: els editors visuals i les llibreries JavaScript.

a) Editors visuals

Els editors visuals són programes amb una interfície gràfica i un seguit de funcions i eines integrades que pretenen facilitar la feina al programador. Si fem la comparació amb l'animació Flash, els editors visuals farien la feina del programa Adobe Flash.

En el cas dels editors d'animacions web HTML5, cal destacar que tots són relativament nous i que cadascun funciona diferent, tot i que mantenen semblances. Tots fan ús de les animacions i transicions CSS3 i alguns també JavaScript, i per la interacció recorren a JavaScript. Els programes no son transparents en quant al codi, és a dir, el programador quan fa l'animació no sap quin codi resultarà.

He analitzat els 4 que m'han semblat més interessants tenint en compte l'empresa que el desenvolupa, la base d'usuaris i la documentació a la xarxa. Els quatre programes són:

- Google Web Designer
- Edge Animator CC
- Tumult Hype 2
- Sencha Animator 1.5.

Google Web Designer

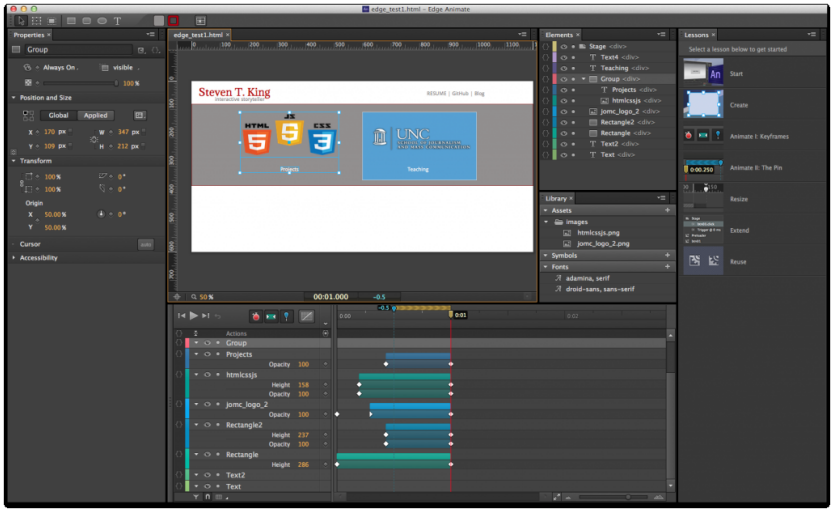


Figura 11: captura del programa Google Web Designer

Es tracta d'un programa gratuït, de moment en fase beta, desenvolupat per Google i multi plataforma (Mac 10.7, Win7, Linux). El mateix programa és una aplicació HTML5, empaquetada usant el "Google Chromium Embedded Framework".

Està destinat a produir anuncis interactius, sobretot per a la plataforma d'anuncis de Google. Incorpora un mode ràpid, per a usuaris bàsics que permet fer bàsicament diapositives, i un mode avançat on veiem una línia de temps i podem crear fotogrames.

Aquest programa és difícil de fer servir. La interfície no és gens intuïtiva i per arribar a crear fotogrames has d'invertir un temps considerable. Incorpora eines d'il·lustració així com també permet importar dibuixos, vídeos i àudio però no té la funció de previsualització directa, com si tenen els altres. Per altra banda, és l'únic dels 4 programes que inclou transformacions 3D, una característica no suportada encara per tots els navegadors.

L'arxiu exportat és directament l'HTML, la qual cosa et permet canviar entre les dues vistes: el disseny i el codi, i pots editar en les dues. A més, l'arxiu resultant és petit, i les animacions CSS3 estan molt ben optimitzades. Hi ha molt poca documentació sobre el programa, Google està destinant molt pocs esforços a divulgar i potenciar el seu ús. Realment, per sortir de la Beta hauran de millorar-lo molt encara.

Edge Animate CC

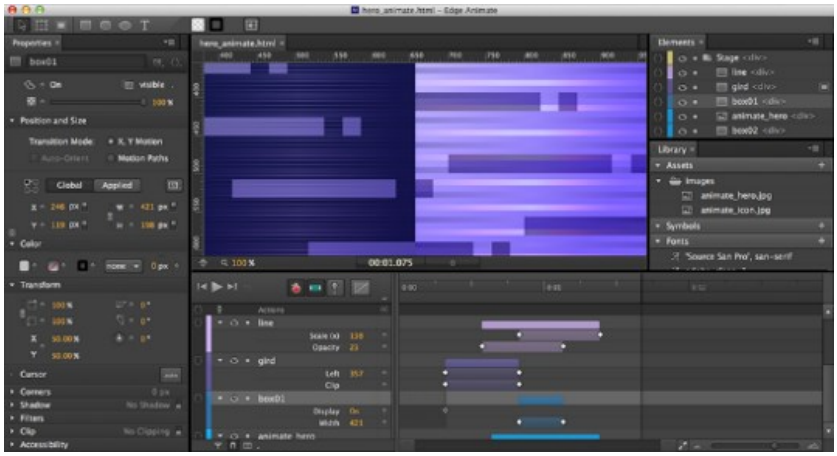


Figura 12: captura del programa Adobe Edge Animate CC

És un programa desenvolupat per l'empresa Adobe. Actualment està disponible per a Windows i Mac i la seva llicència té un preu de 18,44€ al mes.

Edge Animate comparteix molts conceptes amb altres aplicacions d'Adobe, com Flash o After Effects. En quant a les funcionalitats, permet creació d'animacions flexibles, sincronització de l'animació amb àudio i possibilitat d'afegir llibreries JavaScript. També permet la càrrega i creació de plantilles, inclou la possibilitat de copiar-enganxar il·lustracions SVG des d'Illustrator i suporta web font. Ofereix compatibilitat de vídeo HTML5 i inclou una eina per a pujar les animacions resultants als servidors d'Adobe.

Aquest programa fa servir jQuery per a manejar les animacions, i CSS3 per algunes transicions i efectes. Per una banda, això fa que sigui relativament fàcil d'afegir codi extern, per l'altra, per l'altra banda, però, s'està exprimint les funcions jQuery.animate fins als seus límits i això pot causar un baix rendiments en dispositius lents. A part d'això, els arxius de l'animació resultant són més pesats que amb qualsevol dels altres programes d'animació HTML5, la qual cosa resulta en temps de descàrrega més grans.

Tumult Hype 2

Figura 13: captura del programa Tumult Hype 2

Es tracta d'un programa de l'empresa Tumult amb un cost de 29,99 dòlars i disponible només per a ordinadors Mac.

Entre les característiques, destaca la possibilitat de crear múltiples línies de temps. També ens permet crear escenes per separar les diferents animacions, i definir transicions entre les escenes. Podem afegir accions d'interacció, però aleshores hem d'escriure amb codi JavaScript.

En la segona versió inclou Hype Reflect, una funció que permet previsualitzar les creacions directament en els dispositius iOS, a part d'un previsualitzador amb motor Webkit. El punt més dèbil de Tumult és la documentació que ofereix, que és molt pobra si la comparem amb altres programes.

Sencha Animator 1.5

Figura 14: captura del programa Sencha Animator 1.5

Sencha Animator és un programa multi plataforma (Mac, Windows, Linux) de l'empresa Sencha amb un preu de 199 dòlars.

Al web del programa ens prometen que amb ell podem "Crear sofisticades animacions CSS3 amb facilitat". I és que totes les animacions dels elements les fa amb CSS. Ens permet convertir elements i grups d'elements en símbols, semblant al Flash, i crear línies de temps dins d'aquests símbols.

Ofereix moltes animacions predefinides, així com també accions bàsiques de navegació. Si triem JavaScript, ens obre una finestra on afegir codi, i ens ofereix fragments de codi que ens faciliten la feina.

També destaquen el suport de molts tipus d'incrustació, entre les quals Sencha Touch, l'exportació d'animacions compatibles amb totes les xarxes d'anuncis online, el control complet de l'exportació, feedback visual de l'animació en temps real i no es necessita d'escriure ni una línia de codi.

b) Llibreries JavaScript

Les llibreries JavaScript, també anomenades *frameworks*, són fitxers o conjunts de fitxers de codi JavaScript que contenen un seguit de funcions que faciliten molt la programació. En aquest cas, les llibreries que he recercat ens ajuden a desenvolupar animacions HTML5. Per usar-les, només s'ha d'afegir els arxius de la llibreria en l'HTML i fer servir les funcions, objectes, variables, etc. que ens proporcionen.

GSAP

Green Sock Animation Platform (GSAP) és una suite d'eines per a animacions en HTML5 d'alt rendiment. És propietat de l'empresa *GreenSock*, per fer-la servir per a finalitats comercials cal adquirir una llicència. Inclou 4 paquets: *TweenLite*, que és el cor del motor i que gestiona l'animació de gairebé totes les propietats de qualsevol objecte, i 3 altres paquets que afegeixen funcionalitats de gestió de línies de temps i altres.

Les ordres per animar són senzilles. Primer s'instancia l'objecte del motor i després s'escriuen els mètodes d'animació, els quals cadascun inclou l'objecte a animar, la duració, les propietats a animar, la funció que ha de seguir i el temps de retard

Exemple:

```
tl= new TimelineLite();
tl.from($('h2'), 0.8, {opacity:0})
```

GSAP és 20 vegades més ràpid que jQuery i en molts casos més ràpid que les animacions i transicions CSS3, segons els seus creadors. Fa ús de la funció requestAnimationFrame. Utilitza una mateixa sintaxi que serveix per a tots els navegadors. També evita els errors de programari dels navegadors i fa compatibles algunes funcionalitats en navegadors que no les suporten. Aquesta llibreria disposa de molta documentació al web i és utilitzada per molts desenvolupadors webs de referència. És destacable que aquesta llibreria és recomanada als articles de *Google Developers* sobre animació web.

Famo.us

És una llibreria JavaScript de codi obert que inclou un motor de renderitzat 3D que integra animació 3D de físiques, que pot renderitzar al DOM, Canvas o WebGL. És molt nova, va ser presentada el 10 d'abril del 2014. Disposa també de documentació força extensa així com una sèrie de tutorials a la mateixa web oficial.

Raphaël

Raphaël és una llibreria JavaScript per a dibuixar gràfics vectorials amb suport per a molts navegadors. Utilitza SVG i VML per a velles versions d'Internet Explorer. Concretament, aquesta llibreria dona suport a partir de: Chrome 5.0, Firefox 3.0, Safari 3.0, Opera 9.5 i Internet Explorer 6.0. Té llicència MIT, que vol dir que es pot utilitzar gratuïtament. L'autor és Dimitry Baranovskiy.

Snap.svg

És una llibreria JavaScript lliure i gratuïta creada per treballar amb SVG. És molt nova: es va llançar el 23 d'octubre del 2013 i és impulsat per l'empresa Adobe. Aquesta llibreria ha sigut escrita pel mateix autor que Raphaël. El que la diferencia de l'anterior llibreria JavaScript per a SVG, és que està pensada per als navegadors moderns. No és compatible amb navegadors antics, com si que ho és Raphaël, però a canvi ofereix noves funcionalitats com crear màscares, retallar, patrons, degradats, grups. Permet manipular contingut SVG existent i també generar nous elements. Ofereix la càrrega d'arxius SVG mitjançant Ajax.

Hi ha moltes altres llibreries d'animació HTML5 que no son tant utilitzades o bé no són intererssants per als objectius del meu treball, com per exemple *CreateJS*, Velocity.js, *Processing.js*, *Paper.js*, *Kinetic js, Collie js , TaccGL, Animo.js, Move.js, Fabric JS, Three.js, Animate.css* i *Polymer*.

3.3 Producció

Vaig iniciar el desenvolupament d'una animació amb un mètode i una tecnologia triades. A mig fer, però, vaig decidir començar de nou una animació totalment diferent i utilitzant una altra tecnologia. A continuació hi ha l'explicació de la primera opció i de la segona i definitiva.

3.3.1 Primera opció

He dividit el procés de desenvolupament de la primera animació en quatre apartats:

- Tria de la tecnologia
- Disseny i *brieffing*
- Desenvolupament
- Descart

Tria de la tecnologia

El primer pas que vaig fer va ser provar un dels editors visuals. Vaig escollir el programa de Google, el Google Web Designer perquè era el que tenia més a l'abast (és gratuït) i era el que tenia més referències i tutories.

Després d'unes poques proves, la meva valoració va ser força negativa. Sobretot vaig valorar negativament que l'usuari del programa (el desenvolupador de l'animació) té molt poc control sobre el codi. Sí que és veritat que hi ha una opció en la que pots veure i modificar el codi, però resulta que el codi és tant comprimit i confús que és impossible de modificar.

Valoro que tots els elements que dibuixes són transformats a codi, no hi ha imatges. Però quan vaig analitzar el codi em vaig posar les mans al cap: per cada element del dibuix genera una etiqueta canvas i per agrupar-los, els posa dins d'un mateix element "div". Tinc la sensació que aquest programa ha sigut dissenyat només per a usuaris que volen crear el seu propi anunci o baner en HTML5, però no per a desenvolupar una animació més completa o un joc.

L'objectiu del meu treball era estudiar les tecnologies per a fer animacions en HTML5 i posar en pràctica els coneixements en el desenvolupament d'una animació interactiva en HTML5, per la qual cosa vaig pensar que fent servir un editor visual no aprendria les tecnologies, només aprendria a fer anar el programa. Per aquest motiu vaig descartar gairebé des del principi de desenvolupar l'aplicació amb un editor visual.

Un cop descartat de treballar amb un editor visual d'animacions HTML5, vaig començar a fer proves per crear animacions. De la recerca de llibreries per a animacions vaig concloure que GSAP és una de les més noves, actualitzades i amb una extensa base d'usuaris i informació. Vaig assistir a una classe online sobre aquesta llibreria i em va convèncer, així que vaig comprar-me un llibre i vaig decidir començar l'animació amb aquesta eina.

Després d'algunes proves, i de l'experiència que tenia, vaig descartar construir els elements a animar amb HTML + CSS, perquè implica molt i molt temps i el codi resultant és molt extens. Vaig apostar per fer els elements amb un programa de dibuix i exportar-los a PNG.

Disseny i *briefing*

La meva primera idea era animar un àudio-llibre, "El Pere i el Llop". La meva idea va ser crear escenes, fent que l'usuari pugui anar passant entre les escenes com si passés de pàgina.

El públic objectiu de l'animació serien nens de 4-6 anys, igual com l'àudio-llibre. La interacció és tant senzilla, que els nens de 4 anys la poden fer perfectament. Vaig dissenyar i desenvolupar una animació introductòria en la que apareixia el llibre tancat i quan l'usuari clica, s'obre i comença la història. També, vaig aprofitar l'àudio de presentació dels personatges per a fer una escena interactiva en la que quan cliques a cada personatge, se sent la seva presentació.

En quant al disseny, volia que l'animació tingués un aspecte molt simple. Sabia que l'animació resultant seria simple, que es mourien pocs elements; doncs vaig optar per dibuixar només els elements principals.

Per a dibuixar els personatges, em vaig inspirar en els dibuixos de Peter H Reynolds i Turu

Figures 15 i 16: dibuixos de Peter H Reynolds, extrets del lloc web www.peterhreynolds.com

Figures 17 i 18: dibuixos de Turu, extrets del lloc web joanturu.blogspot.com

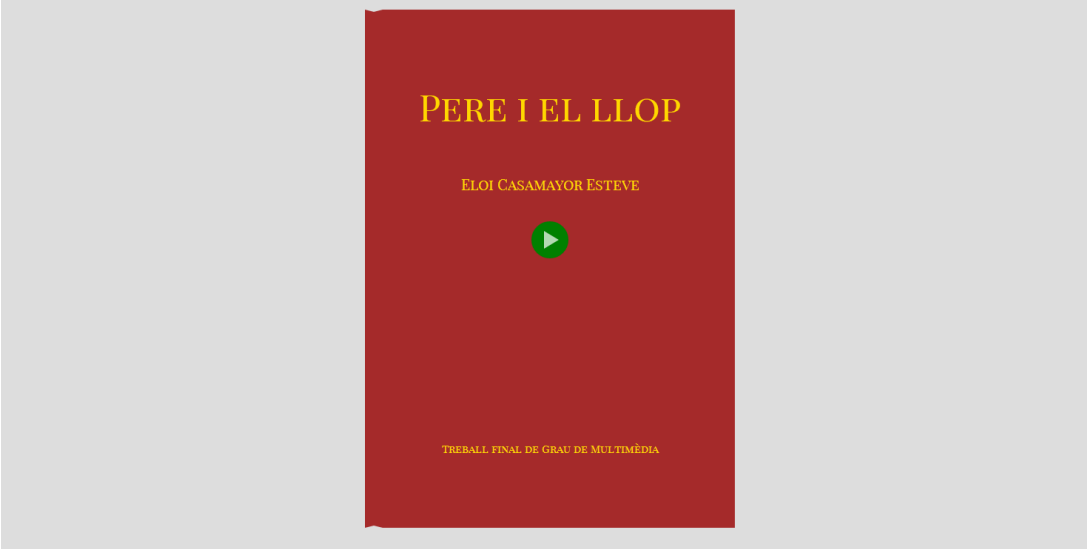


Figura 19: captura de la portada de l'animació "Pere i el Llop"

Figura 20: captura de la introducció de l'animació "Pere i el Llop"

Figura 21: captura de la primera escena de l'animació "Pere i el Llop"

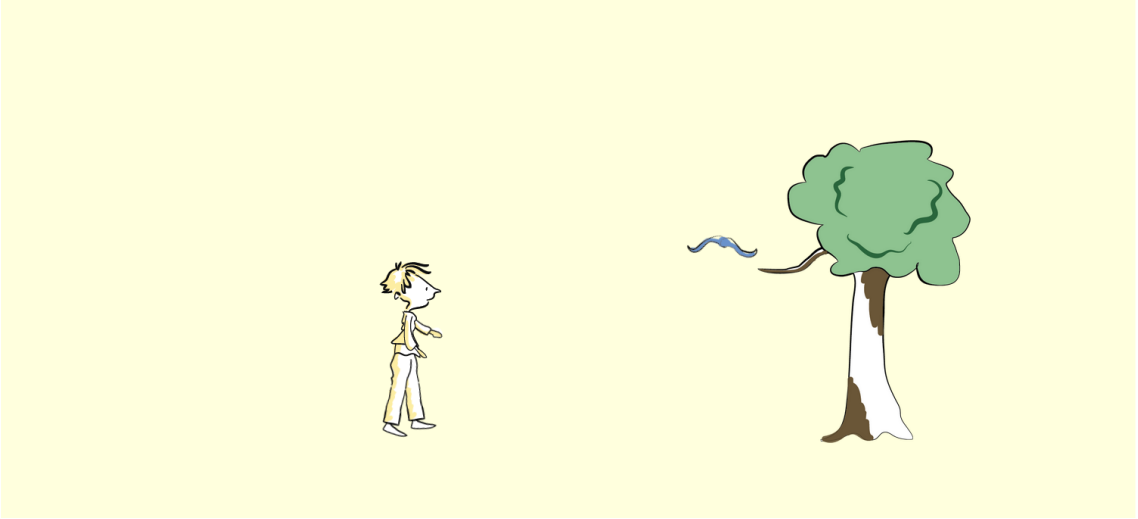


Figura 22: captura de la segona escena de l’animació “Pere i el Llop”

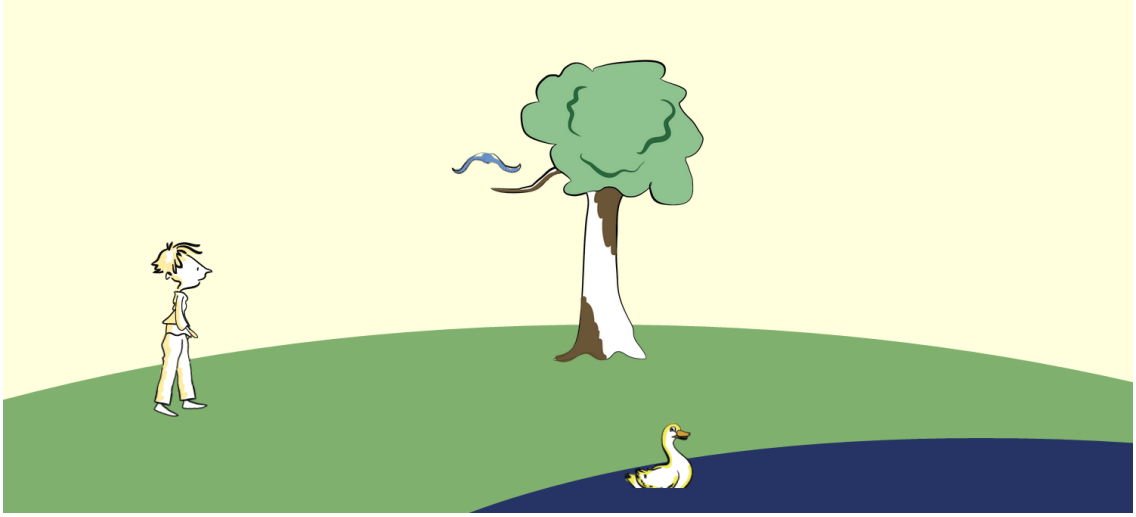


Figura 23: captura de la tercera escena de l’animació “Pere i el Llop”

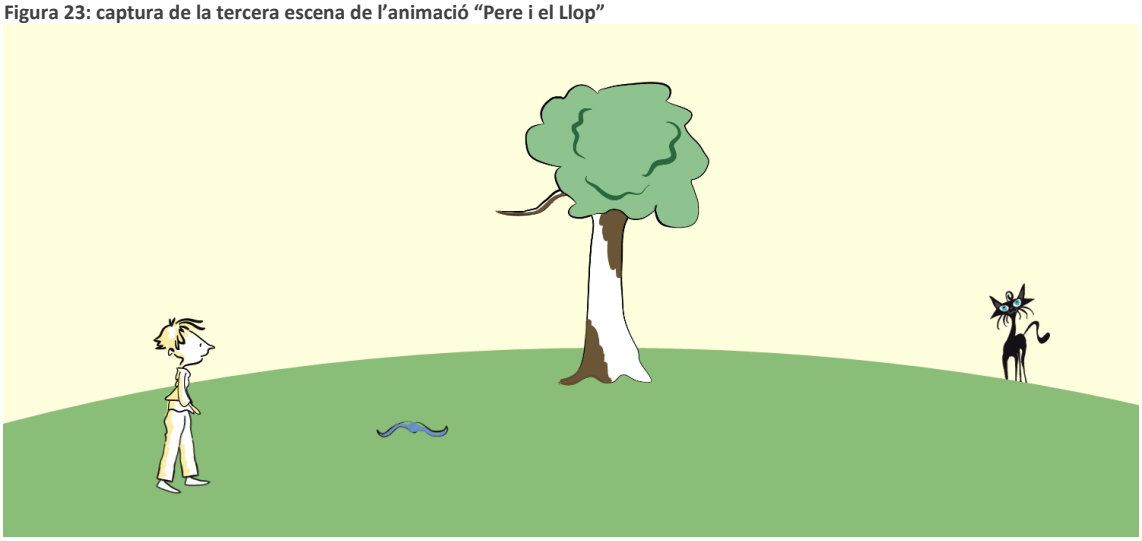


Figura 24: captura de la quarta escena de l’animació “Pere i el Llop”

Desenvolupament

El motor de GSAP està ideat per a quantificar les animacions de translació indicant la quantitat de píxels respecte l'extrem esquerra, el superior, la dreta o l'inferior. Aquest fet fa molt difícil crear una animació que s'adapti a diferents mides de pantalla. Davant d'aquest fet, vaig optar per construir-la dins d'un element div, que es mantenia igual en totes les escenes. Vaig optar per donar a l'element div un estil "*overflow:hidden*", perquè desapareguessin els personatges en sortir del quadre.

He d'afegir que l'última actualització de GSAP, concretament GSAP 1.13.1, del 24 de juliol de 2014, inclou la possibilitat de crear animacions "responsive": que s'adapten a la mida de la pantalla. Per a la meva animació no he pogut gaudir d'aquest avantatge ja que en el moment de llançament de la nova versió ja havia trobat una alternativa.

El desenvolupament de l'animació es basa en anar enllaçant funcions JavaScript que animen els PNG, aprofitant les funcions predefinides de GSAP. L'àudio té molta importància en l'animació, i és clau la sincronització entre aquest i l'animació. Vaig retallar parts de l'àudio perquè en l'original hi havia molts trossos de música que eren espais sense història. Quan s'acaba l'animació d'una escena, apareix l'enllaç que et redirigeix a la pàgina següent, que era un HTML nou.

Vaig comprovar que en mòbils i tauletes, iOS i Android, no es reprodueix un àudio directament al carregar una pàgina, requereix una interacció amb l'usuari per reproduir l'àudio. És per això que vaig provar de no reproduir l'animació fins que l'usuari cliqui al requadre de l'animació.

Descart

Vaig trobar inconvenients que em van fer canviar d'opinió, reenfocar l'animació i començar-la de nou. Un d'ells era la impossibilitat, per la naturalesa de la llibreria GSAP, de fer l'animació *flexible*.

A aquest impediment li donava molta importància, sobretot pensant en la reproducció en mòbils i tauletes. Un altre dels impediments era que el disseny d’animació lineal, seguint l’ordre del conte, donava molt poca llibertat a l’usuari el qual tenia molt poques possibilitats d’interacció. Per últim, tal com l’havia pensat, l’animació es veia pobre, perquè els elements es mouen poc a poc i amb moviments força regulars. En canvi, requeria una inversió d’esforços i de temps molt gran que no podia assumir.

Per tots aquests motius vaig decidir començar de nou redissenyant el concepte.

3.3.2 Funny Circus: opció definitiva

He dividit el procés de desenvolupament de l'animació *Funny Circus* en els següents apartats:

- Tria de la tecnologia
- *Brieffing*
- Disseny dels mini jocs i escenaris
- Punt de vista didàctic
- Desenvolupament

Tria de la tecnologia

Aquesta vegada vaig decidir escollir SVG com a element base del disseny bàsicament perquè, al ser gràfics vectorials, és flexible, pots fer que s'adapti a l'amplada de la pantalla. Aquest element és clau per a la meva animació, perquè vull que sigui adaptable a tot tipus de dispositius. A més a més, em permetia dibuixar tot un escenari amb un programa de dibuix vectorial (en el meu cas, Adobe Illustrator) i exportar-lo a SVG.

En el moment de triar SVG, vaig investigar a fons les maneres d'animar. Va ser llavors quan vaig saber de l'existència de SMIL, i de les etiquetes pròpies de SVG per a animació. Vaig fer algunes proves per a conèixer-ho, però seguidament vaig adonar-me que aquesta tècnica no em bastava per aconseguir tota la interacció i complexitat que desitjava.

Així doncs, després de provar l'animació nativa SVG, vaig seguir apostant per la llibreria GSAP, basada en JavaScript i en la funció RequestAnimationFrame. De seguida em vaig trobar amb un inconvenient: aquesta llibreria no està preparada per animar algunes característiques d'SVG, concretament, la posició dels elements respecte les coordenades SVG. Davant de tal impediment, vaig optar per buscar una altra llibreria per complementar GSAP i poder animar la posició dels objectes.

Vaig trobar la solució al meu problema amb Snap, una llibreria JavaScript oberta per gestionar SVG, impulsada per Adobe i creada pel desenvolupador de l'antiga llibreria Raphaël. Vaig fer proves i després de veure com funcionava, la vaig incloure en les meves eines de treball.

Així doncs, he aprofitat els avantatges de les dues llibreries. De GSAP, he aprofitat la gran facilitat per a crear línies de temps d'animació totalment manejables. De Snap, la possibilitat de carregar arxius SVG mitjançant Ajax simplement amb la funció Snap.load() i la possibilitat d'animar la posició d'elements SVG respecte les coordenades SVG.

Briefing

Vaig definir el públic objectiu del producte en nens i nenes de 6 a 8 anys. També vaig definir un públic objectiu de dispositiu, és a dir el dispositiu en què he pensat perquè sigui reproduïda l'animació, un iPad o tauleta Android de 7 o 9 polsades.

Volia una animació molt més interactiva, on l'usuari tingués molt més de pes. Per això vaig decidir crear una sèrie de mini-jocs en els quals l'usuari ajuda a un personatge a fer els espectacles del circ.

En quant al disseny dels escenaris i del personatge, vaig inspirar-me amb imatges que he trobat per la xarxa.

Figures 25 i 26: Captura de l'animació "Character Animation in Pure CSS", extret del web http://codepen.io/aakashrodrigues/pen/Gfhjw

Disseny de nivells i escenaris

- *Disseny de la navegació i usabilitat*

Per moure's pels diferents escenaris de l'aplicació cal una navegació interna, perquè, al utilitzar Ajax, tota l'aplicació és el mateix arxiu HTML, la qual cosa anul·la la navegació del navegador. Per això he dissenyat uns botons, situats a la part de dalt de la pantalla, per navegar entre els jocs i evitar que l'usuari utilitzi el botó "enrere" del navegador.

Quan estàs dins d'un joc apareix la fletxa cap enrere, a la part superior esquerra, que en clicar-la tornes al menú principal. I quan ets a l'escenari de vestir-se, apareix un botó *done*, a la part superior dreta, el qual quan el cliques vas a parar també al menú principal. També he dissenyat dos botons, situats a dalt a la dreta, per a activar la pantalla completa i per activar la música de fons. Aquests botons hi son sempre i un cop clicats canvien la icona per fer l'acció contrària (desactivar pantalla completa i parar l'àudio).

- *Introducció*

És una animació d'uns 10 segons en la qual l'usuari entra al circ. Apareix un text que diu: *Welcome to Funny Circus, where a man does everything*. Hi ha un botó a dalt a la dreta amb el text *skip intro* el qual, en ser clicat, es para l'animació i apareix l'escenari del menú principal.

Figura 27: Captura de introducció de l'aplicació *Funny Circus*

- *Menú principal*

És un escenari molt simple on l'usuari pot triar entre vestir el personatge o jugar a un dels 4 mini-jocs. Quan cliques a play apareixen les icones que identifiquen cada joc.

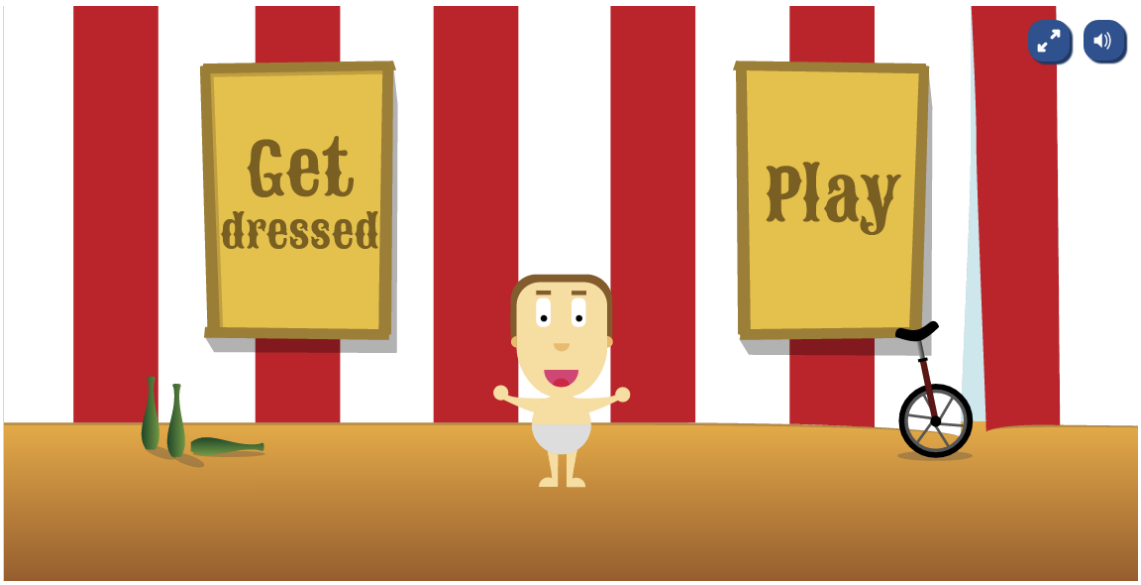


Figura 28: Captura del menú principal de l'aplicació *Funny Circus*

- *Vestidor*

En aquest escenari pots vestir el personatge. Quan cliques sobre els calaixos de la calaixera, s'obren. I arrossegant les peces de roba sobre el personatge el vesteixes. Quan has acabat, cliques el botó amb el text "*done*".

Figura 29: Captura de l'escenari vestidor de l'aplicació *Funny Circus*

- *Joc del Pallasso*

Hem d'ajudar al pallasso a endreçar les seves andròmines. A un costat hi ha els objectes i a l'altre un lloc on guardar-los. Cada objecte té el seu lloc. Simplement arrossegant els objectes al lloc que li correspon aconseguim completar el joc.

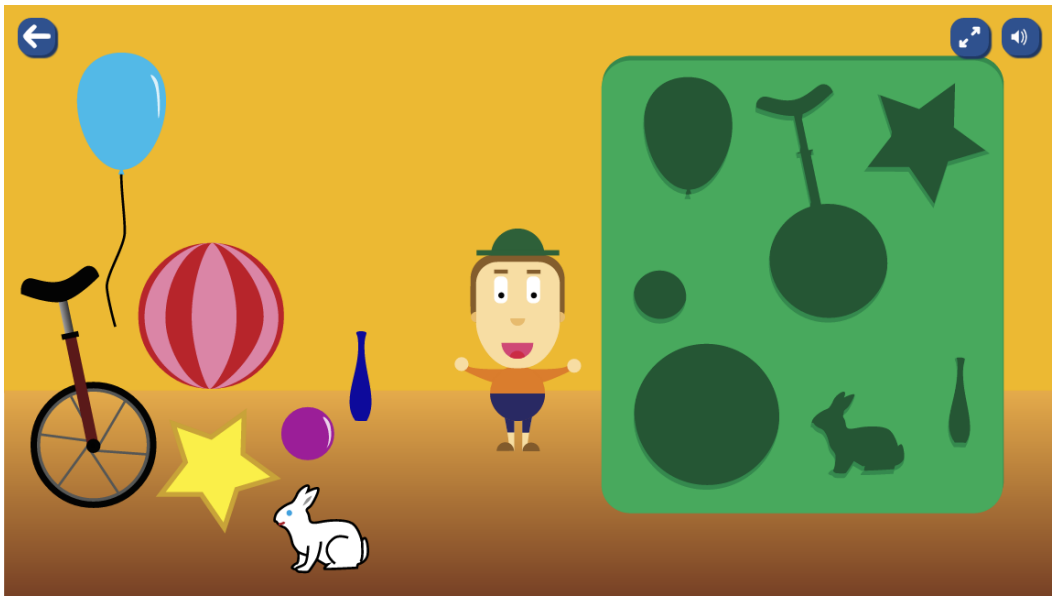


Figura 30: Captura del mini joc del pallasso de l'aplicació *Funny Circus*

- *Joc del domador*

Es tracta d'unir els punts per ordre dels números associats. Quan ha acabat has guanyat, i el lleó es pinta i comença l'animació en què i s'escapa per la dreta.

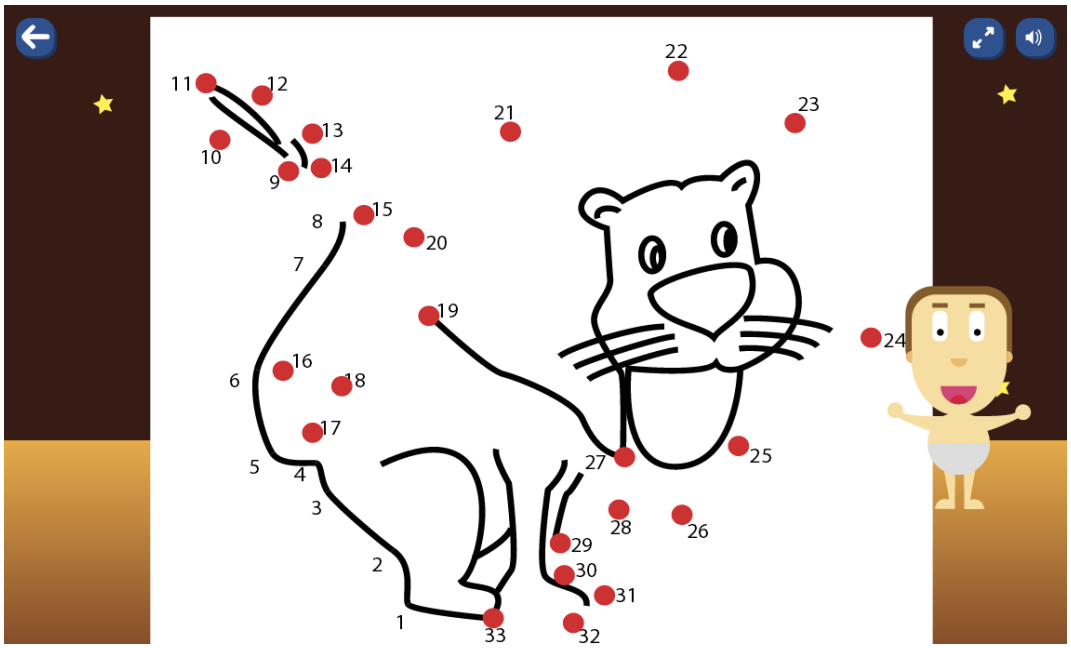


Figura 31: Captura del mini joc del domador de l'aplicació *Funny Circus*

- *Joc de l'equilibrista*

El personatge puja a una barra d'equilibris. L'usuari ha de clicar a banda i banda del personatge per evitar que caigui de la barra. Quan has aguantat uns segons, apareix un botó i si el cliques, el noi salta i has guanyat. El temps que has d'aguantar està representat per una línia verda a dalt de tot.

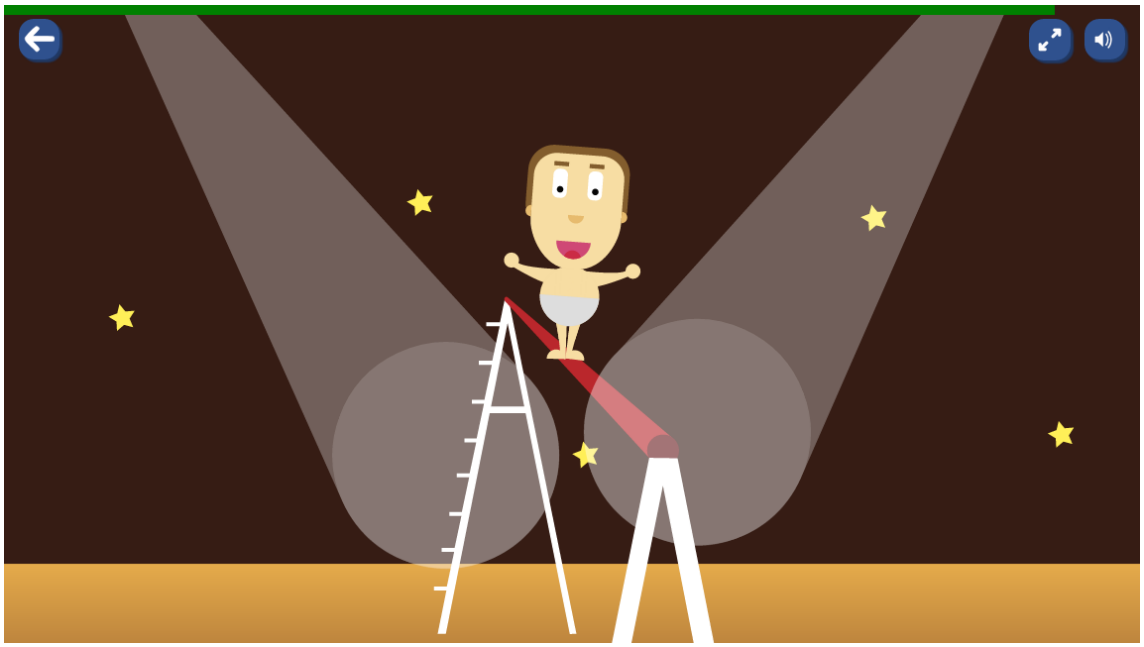


Figura 32: Captura del mini joc de l'equilibrista de l'aplicació *SVG Circus*

- *Joc del malabarista*

El personatge fa malabars i a la part dreta van apareixent pilotes en diferents posicions que es van fer invisibles. Quan una pilota és clicada, desapareix. Si una pilota desapareix del tot, has perdut i al malabarista li cauen les pilotes. El jugador guanya quan aconsegueixes clicar 15 pilotes.

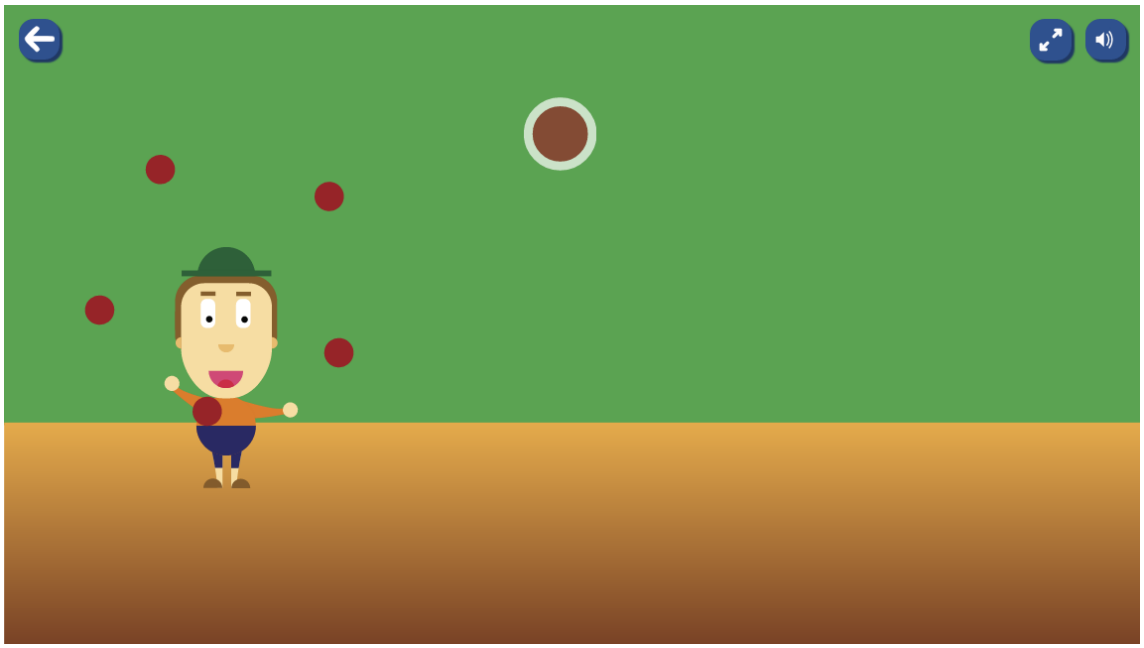


Figura 33: Captura del mini joc del malabarista de l'aplicació *SVG Circus*

Punt de vista didàctic

He realitzat un petit estudi per avaluar l'adequació dels mini jocs al públic objectiu establert i per analitzar els jocs com a eina pedagògica. Per a cada joc, he analitzat els processos cognitius que potencia i els continguts curriculars associats.

Les activitats proposades en els mini jocs, estan dissenyades per al desenvolupament d'habilitats cognitives (memòria, percepció i atenció) amb la finalitat de millorar els processos d'aprenentatge ja que no es pot entendre el pensament i l'aprenentatge sense conèixer els processos cognitius que hi estan implicats. Tanmateix es proposen activitats destinades a reforçar continguts curriculars que afavoriran el seu assoliment.

Pel que fa als processos cognitius, a grans trets cal entendre la memòria com un conjunt multidimensional de processos d'adquisició, retenció i recuperació d'informació i que actua de manera interactiva amb processos perceptius, d'atenció i lingüístics. Per altra banda, la percepció es pot definir com el procés d'extracció activa d'informació, a través dels sentits, que permeten

l'elaboració de representacions. Finalment, l'atenció esdevindria la capacitat de centrar-se en un estímul i ignorar-ne els restants per tal de fer una tasca de manera eficaç.

Pel que fa als continguts curriculars, les activitats s'ajustarien a les competències a assolir durant el Cicle Inicial de l'Educació Primària (de 6 a 8 anys). D'acord amb el currículum vigent, durant el Cicle Inicial s'han de desenvolupar processos específics, entre els quals i en relació a les activitats proposades, destacaríem el de raonament i prova (comparació, diferenciació de regularitats, ordenació, classificació ..) i les connexions (relació, interpretació, aplicació i utilització). Els continguts s'agrupen en tres blocs:

- Numeració i càlcul: Comprensió dels nombres, de les seves formes de representació i del sistema de numeració.
- Relacions i canvis: Comprensió i anàlisi dels patrons, relacions i canvis.
- Espai i forma: Anàlisi de les característiques i propietats de les figures geomètriques. Localització i descripció de relacions espacials. Identificació i aplicació de transformacions geomètriques.

En relació a les activitats proposades s'incidiria especialment ens els següents aspectes:

- Joc del domador: reforçaria el processos de recuperació de la informació així com també la comprensió dels nombres i l'anàlisi dels patrons i formes de representació del sistema de numeració.
- Joc del pallasso: incidiria en la discriminació perceptiva figura-fons així com també en l'anàlisi de les característiques i propietats de les figures geomètriques (reconeixement de figures de dues dimensions en objectes de la realitat)
- Joc de l'equilibrista: Treballaria el reconeixement viso-espacial i la localització i la descripció de relacions espacials. Concretament la interpretació de posicions relatives a l'espai en referència a un mateix i a altres punts.
- Joc del malabarista: Desenvoluparia els processos d'atenció cap a l'estímul presentat (cercle) reforçant la memòria de treball. Tanmateix el reconeixement d'aquesta figura geomètrica.

Finalment cal comentar que la presentació d'aquestes activitats a través del joc afavoreix establir connexions de manera natural i agradable i dominar tècniques bàsiques tant per l'aprenentatge significatiu com per a la memorització.

Desenvolupament

Les animacions les he desenvolupat ajudant-me de les funcions i el sistema d'objectes de les llibreries JavaScript GSAP i Snap, a part de JQuery (que està incorporat a GSAP).

TimelineMax

De GSAP, concretament he optat pel paquet més complet, que és el TimelineMax. Aquest paquet et permet crear més d'una línia de temps que són independents.

Per a crear la línia de temps instancio l'objecte:

```
var tl = new TimelineMax({repeat:2, repeatDelay:1});
```

Per a crear "tweens", que són les ordres que inclou una línia de temps, ho faig de la següent manera:

```
tl.to(mc, 1, {x:100, y:50, alpha:0, delay:2});
```

El ".to" l'utilitzo quan volem assignar la posició a la que volem que arribi l'objecte. També podem usar ".from" si volem que l'objecte parteixi d'aquesta posició i acabi en la posició normal. "mc" és l'objecte que estem animant. Prèviament l'hem hagut de definir, mitjançant jQuery per exemple:

```
mc = $("#maquina")
```

Seguidament s'indica el número de segons que ha de durar l'ordre i finalment s'indica, entre les claus "{}" els paràmetres a animar seguits de dos punts i el valor numèric assignat. Per les distàncies s'utilitza píxels, per al temps segons, per a la rotació graus i per a altres paràmetres de l'1 a 0. Opcionalment, després de la clau "}" es pot indicar la situació temporal que ocupa aquesta ordre. Si no li indiquem, el motor l'executa immediatament després de l'anterior. El paquet TimelineMax inclou ordres per a les línies de temps, com per exemple:

- `tl.pause()` : pausa la línia temporal
- `tl.resume()` : continua l'animació respecte on s'havia pausat
- `tl.seek(3.4)` : continua l'animació des del punt 3.4 segons
- `tl.reverse()` : reprodueix l'animació a l'inrevés.

Càrrega de nivells

Per fer el canvi d'escenari, faig servir la funció de Snap Snap.load(), la qual em permet carregar un objecte SVG del servidor. El fet de carregar els diferents escenaris amb Ajax dóna continuïtat. Les càrregues són molt ràpides, perquè només ha de carregar un arxiu SVG i no tots els arxius JavaScript i l'HTML. Aquesta continuïtat ajuda a l'usuari a sentir-se tota l'estona dins del joc i en quant a programació, serveix per mantenir sempre alguns elements i no haver de carregar-los cada vegada i per poder guardar informació en variables globals, per exemple per vestir el personatge.

Vaig trobar-me amb un error que em sorgia quan carregava per segon cop un dels escenaris. Es creaven dos SVG idèntics. Per solucionar-ho, vaig crear una funció que he afegit al començament de tots els escenaris que comprova que només hi hagi un SVG i si n'hi ha més els elimina. El codi de la funció és aquest:

```
var svgs = document.getElementsByTagName('svg');
if(svgs.length >1) {
svgs[1].remove();
}
```

Construcció del personatge

Vaig crear dos dibuixos del personatge, un de cara i l'altre de costat. El personatge està compost per sis grups d'elements: les 4 extremitats, el cap i el cos. I dins de cada grup hi ha les peces que el formen. En tots els escenaris hi ha els dos dibuixos del personatge, que es van carregant però sempre amb els mateixos elements. Quan m'interessa que estigui de costat faig visible el noi de costat i l'altre invisible, i quan m'interessa que estigui de cara, doncs a l'inrevés.

Perquè sempre vagi vestit com l'hem vestit, vaig crear una funció que s'executa sempre que es carrega un escenari, en un arxiu JavaScript a part, que mira les variables globals i el pinta tal com toca. Les variables adquireixen un número i dins la funció, es recorre un bucle *switch* per a cada variable.

```
function vestirVariables(barret, samarreta, pantalons, sabates){}
```

Creació d'elements "arrossegables"

Per a l'escenari de vestir i per al joc del Pallasso, he fet servir una extensió de GSAP que es diu *Draggable* i que permet convertir qualsevol element en un element que podem arrossegar amb el ratolí (o amb el dit en pantalles tàctils). L'objecte *draggable* conté una matriu amb moltes opcions. Jo he fet servir *dragResistance*, per determinar la resistència que fas a l'arrossegar l'objecte, i *onDragEnd*, que és la funció que executa quan deixes anar l'objecte.

```
Draggable.create("#gorroverd", {
        dragResistance :-0.3,
         onDragEnd:function() {
             if(((this.x<510)&&(this.x>430))&&((this.y>-15)&&(this.y<50))){
               barret=1;
               cabellnormal.attr({opacity:1})
               barretnormal.attr({fill:"#2e6039", opacity:1})
               barretrodo.attr({fill:"rgb(46, 96, 57)", opacity:1})
               visera.attr({fill:"#2e6039", opacity:1})
               barretquadrat.attr({opacity:0})
               cabellpallasso.attr({opacity:0}, pecaposada(1))
               cabellnormal_.attr({opacity:1})
               barretnormal_.attr({fill:"#2e6039", opacity:1})
               barretrodo_.attr({fill:"rgb(46, 96, 57)", opacity:1})
               visera_.attr({fill:"#2e6039", opacity:1})
               barretquadrat_.attr({opacity:0})
               cabellpallasso_.attr({opacity:0})
               tt.to($gorroverd, 0.2, {x:0, y:0, delay:-1.5});
               }else{tt.to($gorroverd, 0.2, {x:0, y:0});}
         }
    });
```

Aconseguir SVG flexible

Si exportes l'SVG amb els valors per defecte d'*Illustrator*, el navegador l'interpreta amb una mida fixa (l'altura i amplada que s'ha definit) i en iOS, es apareix sempre un marge blanc a dalt de l'SVG.

Per fer que l'SVG sigui tant gran com la finestra del navegador i perquè no hi hagi problemes amb iOS, he modificat la capçalera de tots els SVG posant width=100% i preserveAspectRatio="xMinYMin meet".

Exemple de codi d'SVG flexible:

```
<svg version="1.1" id="nom" xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px" width="100%"
viewBox="0 0 1280 720" enable-background="new 0 0 1280 720"
xml:space="preserve" preserveAspectRatio="xMinYMin meet">

</svg>
```

Visualització en pantalla completa

Per aconseguir que l'usuari pugui executar l'aplicació en mode pantalla completa, vaig buscar la funció per fer-ho amb JavaScript i vaig crear un arxiu només per a això. La funció és la següent:

```
function launchFullScreen(element) {
  $("#fullscreen").css("display", "none");
  $("#nofullscreen").css("display", "block");
  if(element.requestFullScreen) {
    element.requestFullScreen();
  } else if(element.mozRequestFullScreen) {
    element.mozRequestFullScreen();
  } else if(element.webkitRequestFullScreen) {
    element.webkitRequestFullScreen();
  }
}
```

Àudio

Vaig comprovar que, malgrat es diu que l'element àudio d'HTML5 és totalment suportat en tots els navegadors, tant d'escriptori com mòbils, no és ben bé així. Primer de tot, en mòbils i tauletes iOS i Android (Windows no ho he provat) no es reprodueix un àudio sense la interacció de l'usuari. Això comporta que per activar una música de fons, li has de demanar a l'usuari que interaccioni d'alguna manera.

A part d'aquest inconvenient, n'hi ha un altre de més gros: Android no és capaç de reproduir dos àudios simultàniament. I iOS tampoc, però en aquest cas no és capaç només quan volem fer play els dos alhora, però si fem play als dos amb un temps entremig, sí que és capaç de reproduir-lo.

Vaig trobar un article d'uns dissenyadors de jocs on s'explica detalladament la problemàtica. Aquests programadors, per solucionar-ho, van crear una petita llibreria JavaScript basada en *sprites* d'àudio anomenada jquery.mb.audio. Tal com es fa habitualment amb imatges, ells proposen enllaçar els àudios (fent que sigui un sol arxiu) i reproduir cada moment que vulguem, el tros d'àudio que necessitem.

He fet servir aquesta llibreria, amb la qual se soluciona el problema de l'àudio en tauletes i mòbils, però a Android la música de fons no es reprodueix.

En aquesta llibreria, es defineix un objecte, anomenat effectSprite, al qual li has de definir la ruta a l'arxiu d'àudio del *sprite* que conté tots els sons. I seguidament, en una matriu s'ha de definir la llista de sons de la matriu, dient el nom, els segons d'inici i final, i les repeticions que volem.

```
effectSprite: {
    id    : "effectSprite",
    ogg   : "sounds/effectsSprite.ogg",
    mp3   : "sounds/effectsSprite.mp3",
    //example of sprite
    sprite: {
        great    : {id: "great", start: 0, end: 1.1, loop: false},
        error    : {id: "error", start: 1.2, end: 1.6, loop: false},
        lion     : {id: "lion", start: 1.6, end: 5.9, loop: false},
        aplausos : {id: "aplausos", start: 6, end: 11.5, loop: false},
        fail     : {id: "fail", start: 11.6, end: 15.8, loop: false},
 }
```

Els arxius que configuren l'aplicació

A continuació anoto esquemàticament tots els arxius que configuren l'aplicació.

```
CIRC>
        svgcircus.html
        svg>
                circ-fora.svg
                main-menu.svg
                getdressed.svg
                joc-equilibrista.svg
                joc-pallasso. Svg
                joc-malabarista.svg
                joc-domador.svg
        sounds>
                bgndsSprite.ogg
                bgndsSprite.mp3
                effectsSprite.ogg
                effectsSprite.mp3
        img>
                fullscreenw.png
                nofullscrenw.png
                audioof.png
                audioon.png
                loading.gif
        js>
                gsap>
                        easing, plugins, utils, jquery i arxius js
                snap>
                        snap.svg.js
                        jquery.mb.audio.js
                        onvestirloaded.js
                        variablesvestir.js
                        fullscreen.js
                        joc1.js
                        joc2.js
                        joc3.js
                joc4.js
                definesounds.js
        css>
                style.css
                back.png
```

Tal com es pot veure en l'esquema anterior, l'aplicació consta només d'un fitxer HTML, amb el nom svgcircus.html. Tots els altres arxius estan agrupats en 5 carpetes: "svg", "sounds", "img", "js" i "css".

En la carpeta "svg" hi ha els arxius amb aquesta extensió, que en són 8: un que dibuixa l'escenari de la introducció, un que dibuixa el menú principal, un altre que dibuixa l'escenari de vestir-se i quatre que dibuixen els escenaris dels mini-jocs.

En la carpeta "sounds" hi ha els arxius d'àudio que es reprodueixen gràcies a la llibreria jquery.mb.audio, una per els efectes de so i una altra per la música de fons, les dues en format .mp3 i .ogg.

En la carpeta "img" hi ha les quatre imatges usades pels botons de navegació.

La carpeta js és la que conté més arxius. Per una banda conté les llibreries Gsap, Snap i jquery.mb.audio. La carpeta Gsap també conté l'arxiu principal de la llibreria jQuery. Per altra banda hi ha els arxius JavaScript que jo mateix he programat. Per organitzar-me millor, he agrupat el codi segons les funcions que fa i quan s'executa. Dins l'arxiu HTML hi ha tot el codi JavaScript que manega la introducció i el menú principal. L'arxiu "onvestirloaded.js" manega l'escenari de vestir-se. Els quatre arxius dels jocs maneguen els quatre mini-jocs. L'arxiu "fullscreen.js" conté les funcions que activen i desactiven la pantalla completa. L'arxiu "definesounds.js" defineix les cadenes de sons que es reprodueixen en els jocs i de fons i l'arxiu "variablesvestir.js" és una única funció que s'executa en carregar cada escenari i vesteix el personatge com correspon, agafant la informació d'unes variables globals definides en l'escenari de vestir-se.

Per últim, la carpeta "css" conté l'arxiu "style.css", que defineix les ordres css, i una imatge utilitzada per un botó de navegació.

Tots els arxius que he creat jo mateix sumen unes 2000 línies de codi JavaScript. Cal puntualitzar que, com que he anat aprenent al llarg del desenvolupament de l'aplicació, el codi no està molt optimitzat. He destinat esforços a optimitzar-lo, però no ha sigut la meva prioritat, ja que el més important era acabar-ho amb èxit.

4. Conclusions

4.1 Avaluació dels objectius

Crec que els objectius que em vaig marcar a l'hora de començar el treball han sigut tots acomplerts amb èxit. Valoro molt positivament la tria que vaig fer dels objectius, tant generals com específics, abans d'endinsar-me en el treball i veure realment què podria assolir.

La feina de recerca i investigació ha sigut fonamental per tal de crear una animació ben feta. Valoro també molt positivament haver estructurat la meva en dues branques, les quals cadascuna m'ha servit molt i l'he aplicat al treball.

La recerca sobre les tecnologies per a realitzar animacions en HTML5 m'ha servit per tenir una visió molt més àmplia i poder triar la tecnologia a fer servir sabent prèviament què podria fer i què no. La recerca de les eines existents, m'ha servit per a descartar el desenvolupament de l'animació amb un editor visual i per trobar unes llibreries que m'han facilitat molt la feina a l'hora d'animar.

L'objectiu principal de crear una animació no només l'he assolit sinó que l'he ampliat: he creat una animació interactiva, un joc. Primerament vaig començar amb la idea de crear una animació lineal d'un conte infantil, però després vaig optar per crear-ne una on l'usuari tingui moltes més possibilitats d'interacció, i he acabat fent uns mini jocs.

Un dels objectius específics, el de crear el guió de l'animació, ha sigut modificat degut a no fer una animació lineal. A part d'un esquema dels escenaris, he hagut de fer un petit estudi sobre la dificultat dels mini jocs, ja que estan destinats a un públic concret.

4.2 L'estat de l'animació web

Fa un temps el web era només contingut. No només era el més important, el contingut, sinó que era l'únic element. El web ha evolucionat molt des dels seus inicis cap als anys 70. Ara mateix, les pàgines web no són només contingut sinó són aplicacions per connectar amb altra gent, per expressar-se, per divertir-se...

Crec que el paper de l'animació web és ajudar a la interacció entre l'usuari i l'ordinador o dispositiu. En la meva opinió, l'animació ha de ser una branca més del disseny gràfic. Per tant, li dono molta importància.

L'animació web avui en dia es troba en un estat força primari. Està en plena evolució, una evolució que és molt ràpida i en la qual hi participen molts actors, però avui en dia, en la meva opinió és encara molt pobra. Si comparem l'animació web amb animacions per a altres plataformes com les animacions natives d'iOS i Android, veiem que aquestes altres estan molt més avançades.

CSS, que és el llenguatge encarregat donar estil a l'HTML5, no està a l'altura de les animacions web. Són lineals per naturalesa, i no ens permeten enllaçar-les

Tal com apunta Marc Grabanski en l'article "HTML5 Needs a New Animation API to Compete with Native", crec que el que fa falta és una nova API (Interfície de programació d'aplicacions) que s'encarregui de gestionar les animacions web.

Les llibreries JavaScript que existeixen, com GSAP, aconsegueixen un gran rendiment, però si el mateix navegador pogués calcular les animacions, com fa amb el CSS, s'aconseguirien uns resultats molt millors.

Hi ha un grup de programadors que ha començat a treballar en aquesta direcció, començant a crear una especificació sobre animació web que pretén englobar tots els recursos que hi ha fins ara. Hi estic interessat i estaré atent a la seva evolució.

5. Agraïments

Vull agrair l’ajuda desinteressada de l’Àngels Pintatella en la part de l’estudi pedagògic. Ella, que és mestra d’escola i coordinadora de cicle infantil, m’ha aconsellat sobre on buscar la informació, m’ha orientat i m’ha ajudat a comprendre l’aspecte pedagògic que hi ha darrere els mini jocs que he creat.

També vull agrair a Jack Doyle les aportacions que m’ha fet a través de correu electrònic. Ell, com a desenvolupador web i com a creador de GSAP, m’ha indicat sobre com buscar entre la documentació de la llibreria i m’ha indicat on buscar inspiració en webs pioneres en l’àmbit de l’animació HTML5.

6. Bibliografia

» Paul Lewis i Paul Irish (7 novembre 2013) "High-performance-animations" [En línia]. Pàgina Web, URL http://www.html5rocks.com/en/tutorials/speed/high-performance-animations/

» Brendan Kenny (22 maig 2014) "Web Animations - element.animate() is now in Chrome 36" [En línia]. Pàgina Web, URL http://updates.html5rocks.com/2014/05/Web-Animations---element-animate-is-now-in-Chrome-36

» Brian Birtles, Shane Stephens, Alex Danilo, Tab Atkin (26 juny 2014) "Web Animations 1.0" (W3C Editor's Draft) [En línia]. Pàgina Web, URL http://dev.w3.org/fxtf/web-animations/

» Dean Jackson, David Hyatt, Chris Marrin, L. David Baron (19 novembre 2013) "CSS Transitions (W3C Working Draft)" [En línia] . Pàgina Web, URL http://www.w3.org/TR/css3-transitions/

» Dean Jackson, David Hyatt, Chris Marrin, Sylvain Galineau, L. David Baron (19 febrer 2013) "CSS Animations" (W3C Working Draft) [En línia] . Pàgina Web, URL http://www.w3.org/TR/css3-transitions/

» Erik Dahlström, Patrick Dengler, Anthony Grasso. Chris Lilley, Cameron McCormack, Doug Schepers, Jonathan Watt, Jon Ferraiolo, 藤沢 淳, Dean Jackson (16 agost 2011) "Scalable Vector Graphics (SVG) 1.1 (Second Edition) [En línia] . Pàgina Web, URL http://www.w3.org/TR/SVG/

» Dick Bulterman, Jack Jansen, Pablo Cesar, Sjoerd Mullender, Eric Hyche, Marisa DeMeglio, Julien Quint, Hiroshi Kawamura, Daniel Weck, Xabiel García Pañeda, David Melendi, Samuel Cruz-Lara, Marcin Hanclik, Daniel F. Zucker, Thierry Michel. (01 desembre 2008) "Synchronized Multimedia Integration Language (SMIL 3.0)" [En línia]]. Pàgina Web, URL http://www.w3.org/TR/smil/

» Pew Research Center (gener 2014) "Mobile Technology Fact Sheet" [En línia]. Pàgina Web, URL http://www.pewinternet.org/fact-sheets/mobile-technology-fact-sheet/

» Jack Doyle "Why GSAP" (2010) [En línia] . Pàgina Web, URL http://www.greensock.com/why-gsap/

» Jack Doyle (13 gener 2014) "Myth Busting: CSS Animations vs. JavaScript" [En línia] . Pàgina Web, URL http://css-tricks.com/myth-busting-css-animations-vs-javascript/

» "Why Snap" [En línia]. Pàgina Web, URL http://snapsvg.io/about/

» Matteo Bicocchi, Pietro Polsinelli (13 de març 2013) "Making HTML5 audio actually work on Mobile" [En línia] http://pupunzi.open-lab.com/2013/03/13/making-html5-audio-actually-work-on-mobile/

» Martin Gittins (22 octubre 2013) "HTML5 app smackdown: Which tool is best?" [En línia]. Pàgina Web, URL http://www.webdesignerdepot.com/2013/10/html5-app-smackdown-which-tool-is-best/

» Anthony Calzadilla (3 maig 2010) "Pure CSS3 Spiderman Cartoon w/ jQuery and HTML5 – Look Ma, No Flash!" [En línia]]. Pàgina Web, URL http://www.optimum7.com/internet-marketing/web-development/pure-css3-spiderman-ipad-cartoon-jquery-html5-no-flash.html

» Pedro Ivo Hudson (Gener 2013) "The Good Man" [En línia]. Pàgina Web, URL http://thegoodman.cc/

» Anthony Calzadilla "MADMANIMATION" [En línia] . Pàgina Web, URL http://stuffandnonsense.co.uk/content/demo/madmanimation/

» Tyler Gaw (abril 2010) "The Man From Hollywood" [En línia] . Pàgina Web, URL http://lab.tylergaw.com/themanfromhollywood/

» Solenoir (31 gener 2014) "Dream On" [En línia]. Pàgina Web, URL http://www.soleilnoir.net/dreamon/

» Good Boy Digital (2013) "Run Pixie Run" [En línia]. Pàgina Web URL http://www.goodboydigital.com/runpixierun/

» Marc Grabanski (21 de maig 2014) "HTML5 Needs a New Animation API to Compete with Native" [En línia]. Pàgina Web, URL http://marcgrabanski.com/html5-animation-api/

» Paul Lewis (13 agost 2014) "Animations" [En línia]. Pàgina Web, URL https://developers.google.com/web/fundamentals/look-and-feel/animations/

» Baroody, A (1988) “Pensamiento matemático de los ninos”. Visor. Madrid.

» Generalitat de Catalunya. Departament d’educació (2009) Currículum educació primària.

» Boanda, H. Caparrós, A. (2008) “Processos psicològics bàsics”. UOC. Barcelona

7. Annexos

Juntament amb aquest document, l'entrega del treball consta d'un seguit d'arxius que són el resultat de la feina feta.

Per una banda entrego els arxius que componen l'animació "Funny Circus", arxius que estan explicats detalladament en el subapartat desenvolupament de l'apartat 3.2.2. Per tal de veure correctament l'animació, els arxius han d'estar allotjats a un servidor. En aquest cas, poden ser consultats temporalment en la URL www.tallerlallana.cat/svgcircus.html.

Per una altra banda, entrego els arxius que componen una part de l'animació que vaig desestimar. Aquesta animació està inacabada i conté errors que no vaig solucionar perquè, com ja explico al llarg de la memòria, vaig decidir abandonar-la. En aquest cas l'animació pot ser visualitzada sense necessitat d'estar allotjada en un servidor. Tot i això, està disponible temporalment a la URL www.tallerlallana.cat/conte.

Per últim, dins la carpeta "Annexes" afegeixo originals dels diagrames de Gantt i els arxius en format Adobe Illustrator (AI) del disseny de les dues animacions.